**Network Device SDK (Windows)**

**User Manual**

[1. Introduction 30](#_Toc149054720)

[1.1 Disclaimer 30](#_Toc149054721)

[1.2 Intended Audience 30](#_Toc149054722)

[1.3 Overview 30](#_Toc149054723)

[1.4 Glossary 30](#_Toc149054724)

[1.5 System Requirements 31](#_Toc149054725)

[2. Version Update 31](#_Toc149054726)

[2.1 Release Notes V2.5 31](#_Toc149054727)

[2.2 Release Notes V2.6 32](#_Toc149054728)

[3. Programming Guide 32](#_Toc149054729)

[3.1 Overview 32](#_Toc149054730)

[3.2 Major Process 33](#_Toc149054731)

[3.2.1 Flowchart 33](#_Toc149054732)

[3.2.2 Sample code for NVR or IPC login 34](#_Toc149054733)

[3.2.3 Sample code for VMS login 36](#_Toc149054734)

[3.3 Process of Parameter Configuration Module 39](#_Toc149054735)

[3.3.1 Flowchart 39](#_Toc149054736)

[3.3.2 Sample code for image configuration 39](#_Toc149054737)

[3.4 Process of Playback & Recording Download Module 42](#_Toc149054738)

[3.4.1 Flowchart 42](#_Toc149054739)

[3.4.2 Sample code for recording search and playback by time 42](#_Toc149054740)

[3.4.3 Sample code for recording search and download by time 45](#_Toc149054741)

[3.4.4 Sample code for recording search and download by filename 48](#_Toc149054742)

[3.5 Process of Live View Module 52](#_Toc149054743)

[3.5.1 Flowchart 52](#_Toc149054744)

[3.5.2 Sample code for live view 52](#_Toc149054745)

[3.6 Process of Audio Module 55](#_Toc149054746)

[3.6.1 Process of two-way audio module 55](#_Toc149054747)

[3.6.2 Process of audio forwarding module 58](#_Toc149054748)

[3.7 Process of Maintenance Module 61](#_Toc149054749)

[3.7.1 Flowchart 61](#_Toc149054750)

[3.7.2 Sample code 61](#_Toc149054751)

[3.8 Process of Exception Module 63](#_Toc149054752)

[3.8.1 Flowchart 63](#_Toc149054753)

[3.8.2 Sample code 64](#_Toc149054754)

[3.9 Process of Alarm Module 66](#_Toc149054755)

[3.9.1 Flowchart 66](#_Toc149054756)

[3.9.2 Sample code 66](#_Toc149054757)

[3.10 Process of PTZ Module 69](#_Toc149054758)

[3.10.1 PTZ control process 69](#_Toc149054759)

[3.10.2 Process of preset and patrol module 72](#_Toc149054760)

[3.11 Process of Smart Module 73](#_Toc149054761)

[3.11.1 Alarm process of person module 73](#_Toc149054762)

[3.11.2 Function process of people counting module 81](#_Toc149054763)

[3.11.3 Function process of vehicle module 82](#_Toc149054764)

[3.11.4 Process of structured data (passing persons and vehicles) 88](#_Toc149054765)

[3.11.5 Process of Smart Intrusion Prevention (SIP) 94](#_Toc149054766)

[3.11.6 Function process of fire detection and thermal imaging 96](#_Toc149054767)

[3.11.7 High-rise littering and elevator entrance detection 97](#_Toc149054768)

[3.11.8 People Counting 98](#_Toc149054769)

[4. Basic Interface Definition 99](#_Toc149054770)

[4.1 SDK Initialization 99](#_Toc149054771)

[4.1.1 SDK initialization 99](#_Toc149054772)

[4.1.2 SDK cleanup 99](#_Toc149054773)

[4.2 SDK Local Functions 100](#_Toc149054774)

[4.2.1 Set log size and quantity 100](#_Toc149054775)

[4.2.2 Set log file path 101](#_Toc149054776)

[4.2.3 Set the flag for writing to logs 101](#_Toc149054777)

[4.2.4 Get SDK version 102](#_Toc149054778)

[4.2.5 Get interface error code 102](#_Toc149054779)

[4.2.6 Register the callback function to receive exceptions 102](#_Toc149054780)

[4.2.7 Set the maximum number of decoding channels 103](#_Toc149054781)

[4.2.8 Set keep-alive interval and times 104](#_Toc149054782)

[4.2.9 Set receiving timeout 105](#_Toc149054783)

[4.2.10 Set local listening address 105](#_Toc149054784)

[4.3 Device Discovery 106](#_Toc149054785)

[4.3.1 Register the callback function to discover devices 106](#_Toc149054786)

[4.3.2 Discover devices 107](#_Toc149054787)

[4.4 Configure No Login 107](#_Toc149054788)

[4.4.1 Modify device IP address 107](#_Toc149054789)

[4.5 User Login 108](#_Toc149054790)

[4.5.1 User logs into the device 108](#_Toc149054791)

[4.5.2 User logout 109](#_Toc149054792)

[4.6 Live View 109](#_Toc149054793)

[4.6.1 Live view 109](#_Toc149054794)

[4.6.2 Stop live view 110](#_Toc149054795)

[4.6.3 Start local recording 111](#_Toc149054796)

[4.6.4 Stop local recording 112](#_Toc149054797)

[4.6.5 Get live view URL 112](#_Toc149054798)

[4.6.6 Start live view by URL 113](#_Toc149054799)

[4.6.7 Snapshot in preview 114](#_Toc149054800)

[4.6.8 Snapshot without preview 115](#_Toc149054801)

[4.6.9 Get image data without preview 116](#_Toc149054802)

[4.6.10 Set recording file size 117](#_Toc149054803)

[4.7 Remote Playback 117](#_Toc149054804)

[4.7.1 Search recording 117](#_Toc149054805)

[4.7.2 Get recording days 119](#_Toc149054806)

[4.7.3 Search the recording distribution info of a specified month 120](#_Toc149054807)

[4.7.4 Get playback URL 121](#_Toc149054808)

[4.7.5 Play recording file by URL 122](#_Toc149054809)

[4.7.6 Play recording file by filename 123](#_Toc149054810)

[4.7.7 Play recording file by time 123](#_Toc149054811)

[4.7.8 Playback by frame 124](#_Toc149054812)

[4.7.9 Playback control 124](#_Toc149054813)

[4.7.10 Stop playback 125](#_Toc149054814)

[4.7.11 Download recording file by filename 126](#_Toc149054815)

[4.7.12 Download recording file by time 126](#_Toc149054816)

[4.7.13 Stop downloading recording file 127](#_Toc149054817)

[4.7.14 Search recording start and end times 128](#_Toc149054818)

[4.8 Stream Data Callback 128](#_Toc149054819)

[4.8.1 Register the callback function to receive raw streams 128](#_Toc149054820)

[4.8.2 Register the callback function to receive decoded audio data 129](#_Toc149054821)

[4.8.3 Register the callback function to receive decoded video data 131](#_Toc149054822)

[4.8.4 Register callback function to receive assembled audio data 132](#_Toc149054823)

[4.8.5 Register callback function to receive assembled video data 133](#_Toc149054824)

[4.8.6 Register image callback 134](#_Toc149054825)

[4.9 Video Parameter Configuration 135](#_Toc149054826)

[4.9.1 Get packet loss rate of window 135](#_Toc149054827)

[4.9.2 Reset packet loss rate of window 136](#_Toc149054828)

[4.9.3 Get bit rate of window 136](#_Toc149054829)

[4.9.4 Get frame rate of window 137](#_Toc149054830)

[4.9.5 Get window resolution 137](#_Toc149054831)

[4.9.6 Get image parameters 138](#_Toc149054832)

[4.9.7 Adjust image parameters 138](#_Toc149054833)

[4.9.8 Get encoding format of window 139](#_Toc149054834)

[4.9.9 Generate a keyframe dynamically 139](#_Toc149054835)

[4.9.10 Set digital zoom 140](#_Toc149054836)

[4.9.11 Enable/disable metadata processing 140](#_Toc149054837)

[4.9.12 Set metadata processing parameter 141](#_Toc149054838)

[4.9.13 Set video fluency 141](#_Toc149054839)

[4.9.14 Set image display scale 142](#_Toc149054840)

[4.10 Audio Function 143](#_Toc149054841)

[4.10.1 Start two-way audio 143](#_Toc149054842)

[4.10.2 Stop two-way audio 144](#_Toc149054843)

[4.10.3 Enable audio data forwarding 144](#_Toc149054844)

[4.10.4 Audio data forwarding 145](#_Toc149054845)

[4.10.5 Stop audio data forwarding 146](#_Toc149054846)

[4.10.6 Start audio broadcast 146](#_Toc149054847)

[4.10.7 Stop audio broadcast 147](#_Toc149054848)

[4.10.8 Enable audio broadcast group 147](#_Toc149054849)

[4.10.9 Modify audio broadcast group 148](#_Toc149054850)

[4.10.10 Get channel status in audio broadcast group 148](#_Toc149054851)

[4.10.11 Modify channel status in audio broadcast group 149](#_Toc149054852)

[4.10.12 Turn on speaker sound 150](#_Toc149054853)

[4.10.13 Get speaker volume 150](#_Toc149054854)

[4.10.14 Set speaker volume 151](#_Toc149054855)

[4.10.15 Turn off speaker sound 152](#_Toc149054856)

[4.10.16 Set mute status 152](#_Toc149054857)

[4.10.17 Get mute status 153](#_Toc149054858)

[4.10.18 Turn on microphone 153](#_Toc149054859)

[4.10.19 Get microphone volume 154](#_Toc149054860)

[4.10.20 Set microphone volume 155](#_Toc149054861)

[4.10.21 Turn off microphone 155](#_Toc149054862)

[4.11 Play Local Video 156](#_Toc149054863)

[4.11.1 Open local file 156](#_Toc149054864)

[4.11.2 Start to play local file 156](#_Toc149054865)

[4.11.3 Stop playing local file 157](#_Toc149054866)

[4.11.4 Get total duration of local file 157](#_Toc149054867)

[4.12 Fisheye Dewarping 158](#_Toc149054868)

[4.12.1 Determine fisheye stream 158](#_Toc149054869)

[4.12.2 Set 5ePTZ display mode 158](#_Toc149054870)

[4.12.3 Mouse movement mode 159](#_Toc149054871)

[4.12.4 Get fisheye dewarping parameters 160](#_Toc149054872)

[4.12.5 Set mode and mount position 160](#_Toc149054873)

[4.12.6 Get mode and mount position 161](#_Toc149054874)

[4.13 Manual Recording 162](#_Toc149054875)

[4.13.1 Get manual recording status 162](#_Toc149054876)

[4.13.2 Start manual recording 162](#_Toc149054877)

[4.13.3 Stop manual recording 163](#_Toc149054878)

[4.14 PTZ Control 163](#_Toc149054879)

[4.14.1 3D positioning 163](#_Toc149054880)

[4.14.2 3D positioning without preview 164](#_Toc149054881)

[4.14.3 Drag to zoom in 165](#_Toc149054882)

[4.14.4 Drag to zoom in without preview 165](#_Toc149054883)

[4.14.5 Drag to zoom out 166](#_Toc149054884)

[4.14.6 Drag to zoom out without preview 166](#_Toc149054885)

[4.14.7 Drag to zoom in/out 167](#_Toc149054886)

[4.14.8 Drag to zoom in/out without preview 168](#_Toc149054887)

[4.14.9 PTZ control operations 168](#_Toc149054888)

[4.14.10 PTZ control operation without preview 169](#_Toc149054889)

[4.14.11 Get or set auto guard info 170](#_Toc149054890)

[4.14.12 Home position operation 170](#_Toc149054891)

[4.14.13 Home position operation without preview 171](#_Toc149054892)

[4.14.14 Absolute PTZ coordinates movement 172](#_Toc149054893)

[4.14.15 Get PTZ status 172](#_Toc149054894)

[4.14.16 Get recorded patrol route 173](#_Toc149054895)

[4.14.17 Get recorded patrol route (including patrol route ID) 173](#_Toc149054896)

[4.14.18 Get patrol status of specified channel 174](#_Toc149054897)

[4.14.19 Recorded patrol operation 174](#_Toc149054898)

[4.14.20 Recorded patrol operation without preview 175](#_Toc149054899)

[4.14.21 Get preset position list 176](#_Toc149054900)

[4.14.22 Preset operation 176](#_Toc149054901)

[4.14.23 Preset operation without preview 177](#_Toc149054902)

[4.14.24 Preset patrol operation 178](#_Toc149054903)

[4.14.25 Preset patrol operation without preview 178](#_Toc149054904)

[4.14.26 Get preset patrol route 179](#_Toc149054905)

[4.14.27 PTZ calibration 180](#_Toc149054906)

[4.14.28 Get current lens angle 180](#_Toc149054907)

[4.14.29 Get PTZ latitude and longitude info 181](#_Toc149054908)

[4.14.30 Set PTZ latitude and longitude info 181](#_Toc149054909)

[4.14.31 Get current zoom ratio 182](#_Toc149054910)

[4.14.32 Set zoom ratio 182](#_Toc149054911)

[4.15 System Configuration 183](#_Toc149054912)

[4.15.1 Enable/disable Telnet 183](#_Toc149054913)

[4.15.2 Export configuration file 184](#_Toc149054914)

[4.15.3 Import configuration file 184](#_Toc149054915)

[4.15.4 Get device configuration info 185](#_Toc149054916)

[4.15.5 Set device configuration info 185](#_Toc149054917)

[4.15.6 Get disk mode 186](#_Toc149054918)

[4.15.7 Set disk mode 187](#_Toc149054919)

[4.15.8 Get system time of device 187](#_Toc149054920)

[4.15.9 Set system time of device 188](#_Toc149054921)

[4.15.10 Modify device IP address without login 188](#_Toc149054922)

[4.15.11 Change device name 189](#_Toc149054923)

[4.15.12 Get device capabilities 189](#_Toc149054924)

[4.15.13 Get device list by device type 190](#_Toc149054925)

[4.15.14 Get device info one by one 191](#_Toc149054926)

[4.15.15 Stop searching device info, release resource 191](#_Toc149054927)

[4.15.16 Get device channel list 192](#_Toc149054928)

[4.15.17 Get device channel basic info one bye one 192](#_Toc149054929)

[4.15.18 Stop getting basic info about device channel 193](#_Toc149054930)

[4.15.19 Get channel list by device ID or channel type 193](#_Toc149054931)

[4.15.20 Get device channel info one by one 194](#_Toc149054932)

[4.15.21 Stop searching device channel info, release resource 194](#_Toc149054933)

[4.15.22 Get detailed channel info by channel type and channel ID 195](#_Toc149054934)

[4.15.23 Set alarm output channel info by channel type and channel ID 195](#_Toc149054935)

[4.15.24 Get channel type 196](#_Toc149054936)

[4.15.25 Get device channel basic info 197](#_Toc149054937)

[4.15.26 Get basic device info 197](#_Toc149054938)

[4.15.27 Get device info 198](#_Toc149054939)

[4.15.28 Get detailed device info 198](#_Toc149054940)

[4.15.29 Get device location info 199](#_Toc149054941)

[4.15.30 Get Wi-Fi sniffer’s MAC address info 199](#_Toc149054942)

[4.15.31 Get video channel list 200](#_Toc149054943)

[4.15.32 Get video channel list 200](#_Toc149054944)

[4.15.33 Get video channel info 201](#_Toc149054945)

[4.15.34 Import audio file 201](#_Toc149054946)

[4.15.35 Delete audio file 202](#_Toc149054947)

[4.15.36 Get audio file 203](#_Toc149054948)

[4.15.37 Get audio output parameters 204](#_Toc149054949)

[4.15.38 Set audio output parameters 205](#_Toc149054950)

[4.16 Alarm-Triggered Snapshot 206](#_Toc149054951)

[4.16.1 Get alarm-triggered snapshot URL 206](#_Toc149054952)

[4.16.2 Save alarm snapshot 208](#_Toc149054953)

[4.17 Alarm Message 208](#_Toc149054954)

[4.17.1 Register the callback function to receive alarm messages 208](#_Toc149054955)

[4.17.2 Register the callback function to receive alarm messages 209](#_Toc149054956)

[4.17.3 Get device alarm info proactively 210](#_Toc149054957)

[4.18 People Flow Counting 211](#_Toc149054958)

[4.18.1 Get people counting list 211](#_Toc149054959)

[4.18.2 Get people counting info one by one 211](#_Toc149054960)

[4.18.3 Stop the search and release resource 212](#_Toc149054961)

[4.18.4 Get people counting statistics 212](#_Toc149054962)

[4.18.5 Reset people counting statistics 213](#_Toc149054963)

[4.18.6 Get time interval of reporting people counting statistics 213](#_Toc149054964)

[4.18.7 Start getting people counting statistics 214](#_Toc149054965)

[4.18.8 Stop getting people counting statistics 215](#_Toc149054966)

[4.18.9 Start getting multi-channel people counting statistics 215](#_Toc149054967)

[4.18.10 Get people counting progress 216](#_Toc149054968)

[4.18.11 Register the callback function to receive people counting statistics 216](#_Toc149054969)

[4.19 Monitoring Task 217](#_Toc149054970)

[4.19.1 Add person 217](#_Toc149054971)

[4.19.2 Delete person 218](#_Toc149054972)

[4.19.3 Add a face monitoring task 219](#_Toc149054973)

[4.19.4 Delete a face monitoring task 219](#_Toc149054974)

[4.19.5 Delete face monitoring tasks in batches 220](#_Toc149054975)

[4.19.6 Create person library info 220](#_Toc149054976)

[4.19.7 Modify person library info 221](#_Toc149054977)

[4.19.8 Delete a person library 221](#_Toc149054978)

[4.19.9 Get configuration of a face monitoring task 222](#_Toc149054979)

[4.19.10 Set a face monitoring task 223](#_Toc149054980)

[4.19.11 Get capacity info of all person libraries 223](#_Toc149054981)

[4.19.12 Search person info with criteria 224](#_Toc149054982)

[4.19.13 Get all the existing person libraries 226](#_Toc149054983)

[4.19.14 Search all the face monitoring tasks 228](#_Toc149054984)

[4.19.15 Add a vehicle library 230](#_Toc149054985)

[4.19.16 Delete a vehicle library 231](#_Toc149054986)

[4.19.17 Modify a vehicle library 231](#_Toc149054987)

[4.19.18 Add vehicle members in batches 232](#_Toc149054988)

[4.19.19 Delete vehicle members in batches 232](#_Toc149054989)

[4.19.20 Delete a vehicle member 233](#_Toc149054990)

[4.19.21 Add a vehicle monitoring task 234](#_Toc149054991)

[4.19.22 Delete vehicle monitoring tasks in batches 234](#_Toc149054992)

[4.19.23 Get detailed info about a vehicle member 235](#_Toc149054993)

[4.19.24 Modify vehicle info in a vehicle library 235](#_Toc149054994)

[4.19.25 Get configuration of a vehicle monitoring task 236](#_Toc149054995)

[4.19.26 Set a vehicle monitoring task 237](#_Toc149054996)

[4.19.27 Assign vehicle members in batches to a vehicle library 237](#_Toc149054997)

[4.19.28 Remove vehicle members in batches from a vehicle library 238](#_Toc149054998)

[4.19.29 Get vehicle image info of a vehicle recognition record 239](#_Toc149054999)

[4.19.30 Get vehicle library list 239](#_Toc149055000)

[4.19.31 Search vehicle recognition records with criteria 241](#_Toc149055001)

[4.19.32 Get all vehicle monitoring tasks 243](#_Toc149055002)

[4.19.33 Search vehicle members with criteria 245](#_Toc149055003)

[4.19.34 Modify a person 248](#_Toc149055004)

[4.19.35 Delete persons in batches 248](#_Toc149055005)

[4.20 Transparent Transmission of Data 249](#_Toc149055006)

[4.20.1 Create transparent channel 249](#_Toc149055007)

[4.20.2 Send data to device’s serial port through transparent channel 250](#_Toc149055008)

[4.20.3 Disconnect transparent channel 251](#_Toc149055009)

[4.21 User Configuration 251](#_Toc149055010)

[4.21.1 Add user info 251](#_Toc149055011)

[4.21.2 Modify user info 252](#_Toc149055012)

[4.21.3 Delete user info 252](#_Toc149055013)

[4.21.4 Change current user’s password 253](#_Toc149055014)

[4.21.5 Change a specified user’s password 254](#_Toc149055015)

[4.21.6 Get current device password 254](#_Toc149055016)

[4.21.7 Get detailed info of a user 255](#_Toc149055017)

[4.21.8 Get detailed info of a user 255](#_Toc149055018)

[4.22 Organization Management 256](#_Toc149055019)

[4.22.1 Add Organization 256](#_Toc149055020)

[4.22.2 Modify Organization 256](#_Toc149055021)

[4.22.3 Delete organizations in batches 257](#_Toc149055022)

[4.22.4 Modify channel by organization ID 258](#_Toc149055023)

[4.22.5 Get channel by organization ID 258](#_Toc149055024)

[4.22.6 Get channel list by organization ID 259](#_Toc149055025)

[4.22.7 Get organization list 261](#_Toc149055026)

[4.23 Network Configuration 263](#_Toc149055027)

[4.23.1 Get NAT types 263](#_Toc149055028)

[4.23.2 Get mapped port 263](#_Toc149055029)

[4.23.3 Set mapped port 264](#_Toc149055030)

[4.24 Decoding Device 265](#_Toc149055031)

[4.24.1 Get capabilities 265](#_Toc149055032)

[4.24.2 Get the number of local encoding channels 265](#_Toc149055033)

[4.24.3 Get video channel list 266](#_Toc149055034)

[4.24.4 Create video wall configuration 266](#_Toc149055035)

[4.24.5 Modify video wall configuration 267](#_Toc149055036)

[4.24.6 Delete video wall configuration 268](#_Toc149055037)

[4.24.7 Get video wall configuration 268](#_Toc149055038)

[4.24.8 Get configuration of a video wall 270](#_Toc149055039)

[4.24.9 Create a virtual LED 271](#_Toc149055040)

[4.24.10 Delete a virtual LED 271](#_Toc149055041)

[4.24.11 Configure a virtual LED 272](#_Toc149055042)

[4.24.12 Get a virtual LED 273](#_Toc149055043)

[4.24.13 Get virtual LED list 273](#_Toc149055044)

[4.24.14 Create a window 275](#_Toc149055045)

[4.24.15 Create windows in batches 276](#_Toc149055046)

[4.24.16 Modify a window 277](#_Toc149055047)

[4.24.17 Modify windows in batches 277](#_Toc149055048)

[4.24.18 Delete a window 278](#_Toc149055049)

[4.24.19 Magnify/restore a window 279](#_Toc149055050)

[4.24.20 Delete windows in batches 279](#_Toc149055051)

[4.24.21 Get window configuration list 280](#_Toc149055052)

[4.24.22 Get configuration of a window 282](#_Toc149055053)

[4.24.23 Get binding info of single live video service 283](#_Toc149055054)

[4.24.24 Bind a live view service 283](#_Toc149055055)

[4.24.25 Unbind a live view service 284](#_Toc149055056)

[4.24.26 Get live view binding info in the scene 285](#_Toc149055057)

[4.24.27 Start passive decoding 287](#_Toc149055058)

[4.24.28 Send data to passive decoding channel 287](#_Toc149055059)

[4.24.29 Stop passive decoding 288](#_Toc149055060)

[4.24.30 Service stream info 289](#_Toc149055061)

[4.24.31 Get all sequence resources of device 291](#_Toc149055062)

[4.24.32 Add a sequence resource to device 291](#_Toc149055063)

[4.24.33 Modify a sequence resource 292](#_Toc149055064)

[4.24.34 Delete a sequence resource 293](#_Toc149055065)

[4.24.35 Get detailed info about a sequence resource 293](#_Toc149055066)

[4.24.36 Create a scene 294](#_Toc149055067)

[4.24.37 Modify a scene 294](#_Toc149055068)

[4.24.38 Delete a scene 295](#_Toc149055069)

[4.24.39 Get the associated scenes of the current scene 296](#_Toc149055070)

[4.24.40 Switch scene 296](#_Toc149055071)

[4.24.41 Get info about the current scene 297](#_Toc149055072)

[4.24.42 Get scene info 297](#_Toc149055073)

[4.24.43 Get scene configuration list 298](#_Toc149055074)

[4.24.44 Bind sequence resource to the scene 300](#_Toc149055075)

[4.24.45 Modify the sequence resource bound to the scene 301](#_Toc149055076)

[4.24.46 Delete the sequence resource bound to the scene 301](#_Toc149055077)

[4.24.47 Control sequence playing 302](#_Toc149055078)

[4.24.48 Add a scene sequence plan 303](#_Toc149055079)

[4.24.49 Modify a scene sequence plan 303](#_Toc149055080)

[4.24.50 Delete a scene sequence plan 304](#_Toc149055081)

[4.24.51 Get a scene sequence plan 305](#_Toc149055082)

[4.24.52 Get scene sequence plan list 305](#_Toc149055083)

[4.24.53 Control the playing of scene sequence plans 307](#_Toc149055084)

[4.25 Smart Function 308](#_Toc149055085)

[4.25.1 Get system image info 308](#_Toc149055086)

[4.25.2 Manual linkage operations 309](#_Toc149055087)

[4.25.3 Register the callback function to receive plate recognition data 309](#_Toc149055088)

[4.25.4 Register the callback function to receive face recognition alarms 310](#_Toc149055089)

[4.25.5 Register the callback function to receive captured face images 311](#_Toc149055090)

[4.25.6 Register the callback function to receive heatmap data 312](#_Toc149055091)

[4.25.7 Register the callback function to receive face alarms 313](#_Toc149055092)

[4.25.8 Register the callback function to receive structured alarms 314](#_Toc149055093)

[4.25.9 Register the callback function to receive plate recognition alarms 315](#_Toc149055094)

[4.25.10 Subscribe to VCA events 316](#_Toc149055095)

[4.25.11 Cancel VCA event subscription 316](#_Toc149055096)

[4.25.12 Alarm subscription via LAPI 317](#_Toc149055097)

[4.25.13 Cancel alarm subscription via LAPI 317](#_Toc149055098)

[4.25.14 Face recognition record 318](#_Toc149055099)

[4.25.15 Parking lot 321](#_Toc149055100)

[4.25.16 Time template 324](#_Toc149055101)

[4.25.17 Access control 329](#_Toc149055102)

[4.25.18 Object search 351](#_Toc149055103)

[4.25.19 Get data related to an alarm 354](#_Toc149055104)

[4.25.20 Register the callback function to receive alarm images 356](#_Toc149055105)

[4.25.21 People counting 357](#_Toc149055106)

[4.25.22 Thermal Imaging 363](#_Toc149055107)

[4.26 System Maintenance 365](#_Toc149055108)

[4.26.1 Search logs 365](#_Toc149055109)

[4.26.2 Get device alarms 368](#_Toc149055110)

[4.26.3 Get alarm logs 370](#_Toc149055111)

[4.26.4 Restart device 372](#_Toc149055112)

[4.26.5 Restore device to factory defaults 372](#_Toc149055113)

[4.27 Structure Definition 373](#_Toc149055114)

[4.27.1 Structure of device login info 373](#_Toc149055115)

[4.27.2 Structure of security login info 373](#_Toc149055116)

[4.27.3 Structure of timeout 374](#_Toc149055117)

[4.27.4 Structure of discovered device info 374](#_Toc149055118)

[4.27.5 Structure of abnormal player output info 375](#_Toc149055119)

[4.27.6 Structure of device address info 376](#_Toc149055120)

[4.27.7 Structure of live view parameters 377](#_Toc149055121)

[4.27.8 Structure of decoded audio data 377](#_Toc149055122)

[4.27.9 Structure of audio parameters 378](#_Toc149055123)

[4.27.10 Structure of channel operation info 378](#_Toc149055124)

[4.27.11 Structure of channel list 379](#_Toc149055125)

[4.27.12 Structure of info about batch processing audio broadcast group 379](#_Toc149055126)

[4.27.13 Structure of info about audio broadcast channels 379](#_Toc149055127)

[4.27.14 Structure of info about audio broadcast group channels 380](#_Toc149055128)

[4.27.15 Structure of info about audio broadcast group control 380](#_Toc149055129)

[4.27.16 Structure of audio data 380](#_Toc149055130)

[4.27.17 Structure of image data 381](#_Toc149055131)

[4.27.18 Structure of assembled audio data 381](#_Toc149055132)

[4.27.19 Structure of parsed video data 382](#_Toc149055133)

[4.27.20 Structure of image info 383](#_Toc149055134)

[4.27.21 Structure of rectangular area 383](#_Toc149055135)

[4.27.22 Structure of recording search info 384](#_Toc149055136)

[4.27.23 Structure of recording file info 384](#_Toc149055137)

[4.27.24 Structure of info about searching recording location by month 385](#_Toc149055138)

[4.27.25 Structure of recording status by month 385](#_Toc149055139)

[4.27.26 Structure of channel list 386](#_Toc149055140)

[4.27.27 Structure of playback by time parameters 386](#_Toc149055141)

[4.27.28 Structure of playback by filename parameters 388](#_Toc149055142)

[4.27.29 Structure of drag-to-zoom 388](#_Toc149055143)

[4.27.30 Structure of info about PTZ areas 389](#_Toc149055144)

[4.27.31 Structure of info about PTZ auto-guard 390](#_Toc149055145)

[4.27.32 Absolute coordinates movement 390](#_Toc149055146)

[4.27.33 PTZ status 391](#_Toc149055147)

[4.27.34 Structure of PTZ recorded patrol route 391](#_Toc149055148)

[4.27.35 PTZ recorded patrol route list 391](#_Toc149055149)

[4.27.36 Structure of PTZ recorded patrol route 392](#_Toc149055150)

[4.27.37 Structure of info about status of PTZ recorded patrol 392](#_Toc149055151)

[4.27.38 Structure of all PTZ presets 393](#_Toc149055152)

[4.27.39 Structure of PTZ preset info 393](#_Toc149055153)

[4.27.40 Structure of detailed info about PTZ preset patrol routes 393](#_Toc149055154)

[4.27.41 Structure of PTZ presets in preset patrol route 394](#_Toc149055155)

[4.27.42 Structure of PTZ preset patrol route list 394](#_Toc149055156)

[4.27.43 Structure of PTZ direction info 395](#_Toc149055157)

[4.27.44 Structure of PTZ lens angle 395](#_Toc149055158)

[4.27.45 PTZ latitude and longitude info unit 395](#_Toc149055159)

[4.27.46 Structure of log search criteria 396](#_Toc149055160)

[4.27.47 Structure of log info 396](#_Toc149055161)

[4.27.48 Structure of alarm info search criteria 397](#_Toc149055162)

[4.27.49 Structure of alarm info 398](#_Toc149055163)

[4.27.50 Alarm log search criteria list 398](#_Toc149055164)

[4.27.51 Structure of alarm log search criteria 399](#_Toc149055165)

[4.27.52 Structure of alarm log info 399](#_Toc149055166)

[4.27.53 Structure of person list 400](#_Toc149055167)

[4.27.54 Structure of person info 401](#_Toc149055168)

[4.27.55 Structure of member’s region info 402](#_Toc149055169)

[4.27.56 Structure of time template info 403](#_Toc149055170)

[4.27.57 Structure of member’s ID info 403](#_Toc149055171)

[4.27.58 Structure of face image list 403](#_Toc149055172)

[4.27.59 Structure of file info 404](#_Toc149055173)

[4.27.60 Structure of custom attribute info 404](#_Toc149055174)

[4.27.61 Structure of staff info 405](#_Toc149055175)

[4.27.62 Structure of visitor info 405](#_Toc149055176)

[4.27.63 Structure of person info result list 406](#_Toc149055177)

[4.27.64 Structure of person info processing result 406](#_Toc149055178)

[4.27.65 Structure of face info processing result 406](#_Toc149055179)

[4.27.66 Structure of monitoring task info 407](#_Toc149055180)

[4.27.67 Structure of info about monitoring task configuration 407](#_Toc149055181)

[4.27.68 Structure of face/vehicle member list 408](#_Toc149055182)

[4.27.69 Structure of alarm linkage configuration info 409](#_Toc149055183)

[4.27.70 Structure of linkage actions list of monitoring task 409](#_Toc149055184)

[4.27.71 Structure of info about linkage action list 410](#_Toc149055185)

[4.27.72 Structure of channel linkage info 410](#_Toc149055186)

[4.27.73 Structure of linkage enablement 410](#_Toc149055187)

[4.27.74 Structure of linked PTZ preset list 411](#_Toc149055188)

[4.27.75 Structure of info about linked PTZ presets 411](#_Toc149055189)

[4.27.76 Structure of linked alarm output 411](#_Toc149055190)

[4.27.77 Structure of logical alarm status of alarm output (manual alarm) 412](#_Toc149055191)

[4.27.78 Structure of schedule configuration (weekly) 412](#_Toc149055192)

[4.27.79 Schedule of schedule configuration (daily) 413](#_Toc149055193)

[4.27.80 Structure of time period configuration 413](#_Toc149055194)

[4.27.81 Structure of arming info 413](#_Toc149055195)

[4.27.82 Structure of monitoring list returned after adding monitoring 414](#_Toc149055196)

[4.27.83 Structure of monitoring info returned after adding monitoring 414](#_Toc149055197)

[4.27.84 Structure of person library info 414](#_Toc149055198)

[4.27.85 Structure of person library list 415](#_Toc149055199)

[4.27.86 Structure of flag for deleting a library 415](#_Toc149055200)

[4.27.87 Structure of info about capacity of all person libraries 416](#_Toc149055201)

[4.27.88 Structure of info about capacity of a face library 416](#_Toc149055202)

[4.27.89 Structure of person info search criteria 417](#_Toc149055203)

[4.27.90 Structure of basic info returned for a batch search 417](#_Toc149055204)

[4.27.91 Structure of list returned for batch deleting face monitoring 418](#_Toc149055205)

[4.27.92 Structure of batch operation info 418](#_Toc149055206)

[4.27.93 Criteria for searching alarm snapshot URL 418](#_Toc149055207)

[4.27.94 Alarm snapshot image info 419](#_Toc149055208)

[4.27.95 Structure of snapshot image info 419](#_Toc149055209)

[4.27.96 Structure of point coordinates 420](#_Toc149055210)

[4.27.97 Structure of manual recording 420](#_Toc149055211)

[4.27.98 Basic device info 421](#_Toc149055212)

[4.27.99 NTP parameters 421](#_Toc149055213)

[4.27.100 Struct of address 422](#_Toc149055214)

[4.27.101 NTP list 422](#_Toc149055215)

[4.27.102 Video stream information of a channel 423](#_Toc149055216)

[4.27.103 Video stream list 423](#_Toc149055217)

[4.27.104 Video stream info 424](#_Toc149055218)

[4.27.105 Video encoding parameter info 424](#_Toc149055219)

[4.27.106 Video collection capability 425](#_Toc149055220)

[4.27.107 Basic attributes of OSD 426](#_Toc149055221)

[4.27.108 OSD time configuration info 426](#_Toc149055222)

[4.27.109 OSD text 427](#_Toc149055223)

[4.27.110 Area 427](#_Toc149055224)

[4.27.111 OSD configuration list 427](#_Toc149055225)

[4.27.112 OSD configuration 428](#_Toc149055226)

[4.27.113 OSD content style 428](#_Toc149055227)

[4.27.114 All alarm input info 429](#_Toc149055228)

[4.27.115 Alarm output info 429](#_Toc149055229)

[4.27.116 Link alarm input 430](#_Toc149055230)

[4.27.117 All alarm input info 430](#_Toc149055231)

[4.27.118 Alarm input info 431](#_Toc149055232)

[4.27.119 Logical alarm status list of alarm output (manual alarm) 431](#_Toc149055233)

[4.27.120 Logical alarm status of alarm output (manual alarm) 431](#_Toc149055234)

[4.27.121 Structure of logical alarm status of triggering or clearing switch output (manual alarm) 432](#_Toc149055235)

[4.27.122 Alarm input configuration list 432](#_Toc149055236)

[4.27.123 Alarm input info 433](#_Toc149055237)

[4.27.124 Image settings 433](#_Toc149055238)

[4.27.125 Image exposure parameters 434](#_Toc149055239)

[4.27.126 Iris info 435](#_Toc149055240)

[4.27.127 Shutter info 435](#_Toc149055241)

[4.27.128 Gain info 436](#_Toc149055242)

[4.27.129 WDR info 436](#_Toc149055243)

[4.27.130 Metering info 437](#_Toc149055244)

[4.27.131 Metering area 437](#_Toc149055245)

[4.27.132 Upper left area 438](#_Toc149055246)

[4.27.133 Lower right area 438](#_Toc149055247)

[4.27.134 Day/night mode info 439](#_Toc149055248)

[4.27.135 Illuminator info 439](#_Toc149055249)

[4.27.136 White balance info 440](#_Toc149055250)

[4.27.137 Network configuration info 441](#_Toc149055251)

[4.27.138 VPN client info 441](#_Toc149055252)

[4.27.139 Device DNS info 443](#_Toc149055253)

[4.27.140 DNS address 443](#_Toc149055254)

[4.27.141 NIC info 444](#_Toc149055255)

[4.27.142 Detailed NIC info 444](#_Toc149055256)

[4.27.143 NIC IPv4 info 445](#_Toc149055257)

[4.27.144 Detailed IPv4 address info 446](#_Toc149055258)

[4.27.145 NIC IPv6 info 446](#_Toc149055259)

[4.27.146 Detailed IPv6 address info 446](#_Toc149055260)

[4.27.147 Port information 447](#_Toc149055261)

[4.27.148 Privacy mask configuration info 447](#_Toc149055262)

[4.27.149 Structure of area configuration 448](#_Toc149055263)

[4.27.150 Tampering detection info 448](#_Toc149055264)

[4.27.151 Motion detection analysis info 449](#_Toc149055265)

[4.27.152 Cross line detection analysis info 449](#_Toc149055266)

[4.27.153 Area info 450](#_Toc149055267)

[4.27.154 Intrusion detection info 450](#_Toc149055268)

[4.27.155 Intrusion detection area info 451](#_Toc149055269)

[4.27.156 Coordinates of intrusion detection area 451](#_Toc149055270)

[4.27.157 HDD list 452](#_Toc149055271)

[4.27.158 Detailed HDD info 452](#_Toc149055272)

[4.27.159 Focus info 453](#_Toc149055273)

[4.27.160 Day/night mode info 453](#_Toc149055274)

[4.27.161 Defog info 453](#_Toc149055275)

[4.27.162 Image enhancement parameters of specified channel 454](#_Toc149055276)

[4.27.163 Info about audio input parameters 454](#_Toc149055277)

[4.27.164 Audio input channel info 455](#_Toc149055278)

[4.27.165 DST configuration info 456](#_Toc149055279)

[4.27.166 DST configuration 456](#_Toc149055280)

[4.27.167 Time configuration 457](#_Toc149055281)

[4.27.168 Recording schedule configuration info 457](#_Toc149055282)

[4.27.169 Recording schedule rules 458](#_Toc149055283)

[4.27.170 Recording schedule (weekly) configuration 458](#_Toc149055284)

[4.27.171 Recording schedule (daily) configuration 459](#_Toc149055285)

[4.27.172 Video time period configuration 459](#_Toc149055286)

[4.27.173 Snapshot info of video input channel 460](#_Toc149055287)

[4.27.174 Video source resolution information 460](#_Toc149055288)

[4.27.175 Schedule (weekly) configuration 461](#_Toc149055289)

[4.27.176 Schedule (daily) configuration 461](#_Toc149055290)

[4.27.177 Time period configuration 462](#_Toc149055291)

[4.27.178 Arming schedule configuration of alarm input/output (weekly) 462](#_Toc149055292)

[4.27.179 Recording status list 462](#_Toc149055293)

[4.27.180 Recording status info 463](#_Toc149055294)

[4.27.181 Serial port info 463](#_Toc149055295)

[4.27.182 Serial port configuration info 464](#_Toc149055296)

[4.27.183 Serial port parameters 464](#_Toc149055297)

[4.27.184 Audio status list 465](#_Toc149055298)

[4.27.185 Audio status info 465](#_Toc149055299)

[4.27.186 Video loss alarm configuration info 466](#_Toc149055300)

[4.27.187 Linkage action list 466](#_Toc149055301)

[4.27.188 Linkage actions of task 466](#_Toc149055302)

[4.27.189 Channel linkage info 467](#_Toc149055303)

[4.27.190 Linkage action enablement parameter 467](#_Toc149055304)

[4.27.191 PTZ preset 468](#_Toc149055305)

[4.27.192 Info about linked PTZ preset 468](#_Toc149055306)

[4.27.193 Alarm output 469](#_Toc149055307)

[4.27.194 Logical alarm status of alarm output (manual alarm) 469](#_Toc149055308)

[4.27.195 Linkage action enablement parameters 469](#_Toc149055309)

[4.27.196 Tampering detection configuration info 470](#_Toc149055310)

[4.27.197 Structure of info about temperature detection blackbody 470](#_Toc149055311)

[4.27.198 Structure of info about temperature detection correction 471](#_Toc149055312)

[4.27.199 Structure of info about temperature detection unit 471](#_Toc149055313)

[4.27.200 Motion detection area type info 471](#_Toc149055314)

[4.27.201 Motion detection grid area info 472](#_Toc149055315)

[4.27.202 Motion detection rectangle area list 472](#_Toc149055316)

[4.27.203 Motion detection rectangle area info 473](#_Toc149055317)

[4.27.204 Coordinates info of rectangle area 473](#_Toc149055318)

[4.27.205 Linkage actions of alarm input/output 474](#_Toc149055319)

[4.27.206 RAID status info 474](#_Toc149055320)

[4.27.207 HDD list 475](#_Toc149055321)

[4.27.208 HDD info 475](#_Toc149055322)

[4.27.209 Storage container list 476](#_Toc149055323)

[4.27.210 Storage container info 477](#_Toc149055324)

[4.27.211 Expansion storage container info 478](#_Toc149055325)

[4.27.212 HDD S.M.A.R.T. info 478](#_Toc149055326)

[4.27.213 HDD S.M.A.R.T. info 479](#_Toc149055327)

[4.27.214 Photo server configuration list 480](#_Toc149055328)

[4.27.215 Enhancement configuration info 480](#_Toc149055329)

[4.27.216 Smart server list 481](#_Toc149055330)

[4.27.217 Smart server configuration info 481](#_Toc149055331)

[4.27.218 Server user info 482](#_Toc149055332)

[4.27.219 Motion detection alarm parameters 482](#_Toc149055333)

[4.27.220 Management server list 483](#_Toc149055334)

[4.27.221 Visible range parameters 483](#_Toc149055335)

[4.27.222 Verification info 484](#_Toc149055336)

[4.27.223 Storage quota info 484](#_Toc149055337)

[4.27.224 Custom protocol info 485](#_Toc149055338)

[4.27.225 Custom protocol configuration info 486](#_Toc149055339)

[4.27.226 Custom stream protocol info 486](#_Toc149055340)

[4.27.227 Simple info of custom protocol 487](#_Toc149055341)

[4.27.228 Custom protocol info 487](#_Toc149055342)

[4.27.229 Storage policy info 488](#_Toc149055343)

[4.27.230 Video storage policy info 488](#_Toc149055344)

[4.27.231 Image storage policy info 489](#_Toc149055345)

[4.27.232 Back focus finetune parameters 489](#_Toc149055346)

[4.27.233 Smart attribute configuration 490](#_Toc149055347)

[4.27.234 Face attribute configuration 490](#_Toc149055348)

[4.27.235 Temperature attribute configuration 490](#_Toc149055349)

[4.27.236 Image correction parameters 491](#_Toc149055350)

[4.27.237 Device time synchronization 491](#_Toc149055351)

[4.27.238 Automatic scene switching 492](#_Toc149055352)

[4.27.239 Scene template info 492](#_Toc149055353)

[4.27.240 All scene info 493](#_Toc149055354)

[4.27.241 Scene info 493](#_Toc149055355)

[4.27.242 Detailed info about automatic scene switching triggers 494](#_Toc149055356)

[4.27.243 Environment parameter info 494](#_Toc149055357)

[4.27.244 Attribute collection info 495](#_Toc149055358)

[4.27.245 Face attributes enable/disable info 495](#_Toc149055359)

[4.27.246 Pedestrian attributes enable/disable info 496](#_Toc149055360)

[4.27.247 Non-motor vehicle attributes enable/disable info 497](#_Toc149055361)

[4.27.248 Motor vehicle attributes enable/disable info 498](#_Toc149055362)

[4.27.249 Time configuration 499](#_Toc149055363)

[4.27.250 Time parameters 499](#_Toc149055364)

[4.27.251 System IP address 500](#_Toc149055365)

[4.27.252 Position relative to video wall 500](#_Toc149055366)

[4.27.253 Coordinates info 501](#_Toc149055367)

[4.27.254 Detailed HDD S.M.A.R.T. info 501](#_Toc149055368)

[4.27.255 Photo server configuration info 502](#_Toc149055369)

[4.27.256 Smart server configuration info 503](#_Toc149055370)

[4.27.257 All OSD contents. 504](#_Toc149055371)

[4.27.258 OSD content info 504](#_Toc149055372)

[4.27.259 Content info 505](#_Toc149055373)

[4.27.260 Temperature detection alarm configuration info 505](#_Toc149055374)

[4.27.261 Structure of basic info about temperature detection 506](#_Toc149055375)

[4.27.262 Temperature detection common rule list 506](#_Toc149055376)

[4.27.263 Structure of general info about temperature detection 507](#_Toc149055377)

[4.27.264 Temperature detection comparison rule list 507](#_Toc149055378)

[4.27.265 Structure of comparison info about temperature detection 508](#_Toc149055379)

[4.27.266 Basic device info 508](#_Toc149055380)

[4.27.267 IP address info 510](#_Toc149055381)

[4.27.268 User account info 511](#_Toc149055382)

[4.27.269 Onvif info 511](#_Toc149055383)

[4.27.270 GB info 512](#_Toc149055384)

[4.27.271 Smart lock info 512](#_Toc149055385)

[4.27.272 Player info 513](#_Toc149055386)

[4.27.273 Resolution information 513](#_Toc149055387)

[4.27.274 Device channel info 514](#_Toc149055388)

[4.27.275 Channel info about Xware device’s local signal source 514](#_Toc149055389)

[4.27.276 Basic channel info of device 515](#_Toc149055390)

[4.27.277 Basic device info 516](#_Toc149055391)

[4.27.278 Time info 517](#_Toc149055392)

[4.27.279 Client version list 517](#_Toc149055393)

[4.27.280 Client version info 518](#_Toc149055394)

[4.27.281 Device info 518](#_Toc149055395)

[4.27.282 Detailed device info 519](#_Toc149055396)

[4.27.283 Device firmware info 519](#_Toc149055397)

[4.27.284 Geolocation info 520](#_Toc149055398)

[4.27.285 Wi-Fi sniffer MAC array info 520](#_Toc149055399)

[4.27.286 Wi-Fi sniffer MAC info 520](#_Toc149055400)

[4.27.287 Detailed video channel info 521](#_Toc149055401)

[4.27.288 Detailed video channel info 521](#_Toc149055402)

[4.27.289 Detailed video channel info (expansion) 523](#_Toc149055403)

[4.27.290 Alarm/event report info 524](#_Toc149055404)

[4.27.291 Alarm message 524](#_Toc149055405)

[4.27.292 Event info 526](#_Toc149055406)

[4.27.293 Event resource info 526](#_Toc149055407)

[4.27.294 Pull alarm info 527](#_Toc149055408)

[4.27.295 People counting 527](#_Toc149055409)

[4.27.296 People counting command 528](#_Toc149055410)

[4.27.297 People counting 528](#_Toc149055411)

[4.27.298 Time interval for reporting people counting statistics 529](#_Toc149055412)

[4.27.299 Multi-channel people counting command 529](#_Toc149055413)

[4.27.300 People counting info 530](#_Toc149055414)

[4.27.301 Structure of info about Xware device capability 530](#_Toc149055415)

[4.27.302 Structure of video wall capability 531](#_Toc149055416)

[4.27.303 Structure of video output format list 532](#_Toc149055417)

[4.27.304 Structure of window capability 532](#_Toc149055418)

[4.27.305 Structure of background image capability 533](#_Toc149055419)

[4.27.306 Structure of virtual LED capability 533](#_Toc149055420)

[4.27.307 Structure of sequence display capability 534](#_Toc149055421)

[4.27.308 Structure of feature capability 535](#_Toc149055422)

[4.27.309 Structure of info about window layout supported by Xware device 536](#_Toc149055423)

[4.27.310 Structure of video output expansion format of Xware device 536](#_Toc149055424)

[4.27.311 Structure of info about the number of channels 536](#_Toc149055425)

[4.27.312 Structure of info about decoding channels 537](#_Toc149055426)

[4.27.313 Structure of info about video input channels 538](#_Toc149055427)

[4.27.314 Structure of info about video output channel of Xware device 538](#_Toc149055428)

[4.27.315 Structure of info about audio input channel 539](#_Toc149055429)

[4.27.316 Structure of info about audio output channel 539](#_Toc149055430)

[4.27.317 Structure of info about serial interface channel 540](#_Toc149055431)

[4.27.318 Structure of video wall info 540](#_Toc149055432)

[4.27.319 Structure of video wall screen configuration 541](#_Toc149055433)

[4.27.320 Structure of special output format specs 542](#_Toc149055434)

[4.27.321 Structure of LED screens 542](#_Toc149055435)

[4.27.322 Structure of physical output configuration 542](#_Toc149055436)

[4.27.323 Structure of virtual LED info 543](#_Toc149055437)

[4.27.324 Structure of background 543](#_Toc149055438)

[4.27.325 Structure of font info 544](#_Toc149055439)

[4.27.326 Structure of window info 545](#_Toc149055440)

[4.27.327 Structure of split window info 546](#_Toc149055441)

[4.27.328 Structure of window list info 546](#_Toc149055442)

[4.27.329 Structure of window info for batch window opening 547](#_Toc149055443)

[4.27.330 Structure of info about batch operation results 547](#_Toc149055444)

[4.27.331 Structure of returned result of opening windows in batches 548](#_Toc149055445)

[4.27.332 Structure of screen zoom info 548](#_Toc149055446)

[4.27.333 Structure of decoding device ID info 549](#_Toc149055447)

[4.27.334 Structure of video source info 549](#_Toc149055448)

[4.27.335 Structure of authentication info 550](#_Toc149055449)

[4.27.336 Structure of font info 550](#_Toc149055450)

[4.27.337 Structure of info about control parameters of sending passive decoding data 551](#_Toc149055451)

[4.27.338 Structure of detailed stream session info 551](#_Toc149055452)

[4.27.339 Structure of basic info about sequence resource 552](#_Toc149055453)

[4.27.340 Structure of detailed info about sequence resource 552](#_Toc149055454)

[4.27.341 Structure of info about video sources of sequence resource 553](#_Toc149055455)

[4.27.342 Structure of basic video source info for remote live video 553](#_Toc149055456)

[4.27.343 Structure of basic scene info 554](#_Toc149055457)

[4.27.344 Structure of scene modification info 555](#_Toc149055458)

[4.27.345 Structure of detailed scene info 555](#_Toc149055459)

[4.27.346 Structure of all background image info 556](#_Toc149055460)

[4.27.347 Structure of sequence resource list 557](#_Toc149055461)

[4.27.348 Structure of sequence resource info 557](#_Toc149055462)

[4.27.349 Structure of brief scene info 557](#_Toc149055463)

[4.27.350 Structure of sequence resource binding info 558](#_Toc149055464)

[4.27.351 Structure of info about sequence play control 558](#_Toc149055465)

[4.27.352 Structure of basic info about scene sequence plan 559](#_Toc149055466)

[4.27.353 Structure of info about scene switching timer 560](#_Toc149055467)

[4.27.354 Structure of info about scene switching time table 560](#_Toc149055468)

[4.27.355 Structure of info about 24-hour time table 560](#_Toc149055469)

[4.27.356 Structure of info about scene sequence plan playing control 561](#_Toc149055470)

[4.27.357 Structure of stream source info 561](#_Toc149055471)

[4.27.358 Structure of stream destination info 562](#_Toc149055472)

[4.27.359 Structure of detailed stream info 562](#_Toc149055473)

[4.27.360 Structure of manual alarm linkage 563](#_Toc149055474)

[4.27.361 Structure of rectangle coordinates 563](#_Toc149055475)

[4.27.362 Structure of plate recognition data info 563](#_Toc149055476)

[4.27.363 Structure of plate recognition xml info 564](#_Toc149055477)

[4.27.364 Structure of plate recognition image info 564](#_Toc149055478)

[4.27.365 Structure of face recognition record 565](#_Toc149055479)

[4.27.366 Structure of info about face snapshot comparison 565](#_Toc149055480)

[4.27.367 Structure of face library member info 566](#_Toc149055481)

[4.27.368 Structure of snapshot image info 567](#_Toc149055482)

[4.27.369 Structure of region info of face library members 567](#_Toc149055483)

[4.27.370 Structure of member’s ID info 568](#_Toc149055484)

[4.27.371 Structure of area coordinates 568](#_Toc149055485)

[4.27.372 Structure of face snapshot info 568](#_Toc149055486)

[4.27.373 Structure of face position info 569](#_Toc149055487)

[4.27.374 Structure of heatmap data info 570](#_Toc149055488)

[4.27.375 Structure of heatmap xml info 570](#_Toc149055489)

[4.27.376 Structure of heatmap area info 571](#_Toc149055490)

[4.27.377 Person alarm info 571](#_Toc149055491)

[4.27.378 Face pass-through records info 572](#_Toc149055492)

[4.27.379 Face comparison info 573](#_Toc149055493)

[4.27.380 Semi-structured attribute info 573](#_Toc149055494)

[4.27.381 Face attribute info 574](#_Toc149055495)

[4.27.382 Person attributes 574](#_Toc149055496)

[4.27.383 Structured alarm info 575](#_Toc149055497)

[4.27.384 Structured data info 576](#_Toc149055498)

[4.27.385 Object info 576](#_Toc149055499)

[4.27.386 Face info 577](#_Toc149055500)

[4.27.387 Person info 578](#_Toc149055501)

[4.27.388 Non-motor vehicle info 579](#_Toc149055502)

[4.27.389 Non-motor vehicle attribute info 579](#_Toc149055503)

[4.27.390 Vehicle info 580](#_Toc149055504)

[4.27.391 Vehicle attribute info 581](#_Toc149055505)

[4.27.392 Plate attribute info 581](#_Toc149055506)

[4.27.393 Structure of vehicle recognition event 582](#_Toc149055507)

[4.27.394 Structure of vehicle comparison alarm info 582](#_Toc149055508)

[4.27.395 Structure of vehicle recognition record info 583](#_Toc149055509)

[4.27.396 Structure of plate info 583](#_Toc149055510)

[4.27.397 Structure of info about smart event subscription 584](#_Toc149055511)

[4.27.398 Structure of smart event info 584](#_Toc149055512)

[4.27.399 Structure of info about LAPI alarm subscription 585](#_Toc149055513)

[4.27.400 Structure of returned result for successful subscriptions 585](#_Toc149055514)

[4.27.401 Structure of returned alarm records (face recognition and plate recognition) 586](#_Toc149055515)

[4.27.402 Structure of parking lot entrance/exit list 586](#_Toc149055516)

[4.27.403 Structure of parking lot entrance/exit info 587](#_Toc149055517)

[4.27.404 Structure of parking lot list 587](#_Toc149055518)

[4.27.405 Structure of parking log info 587](#_Toc149055519)

[4.27.406 Structure of parking lot entrance/exit ID 588](#_Toc149055520)

[4.27.407 Structure of lane list 588](#_Toc149055521)

[4.27.408 Structure of lane info 589](#_Toc149055522)

[4.27.409 Structure of custom time periods 590](#_Toc149055523)

[4.27.410 Structure of vehicle let-through info 590](#_Toc149055524)

[4.27.411 Structure of parking payment order info 590](#_Toc149055525)

[4.27.412 Structure of parking cost 591](#_Toc149055526)

[4.27.413 Structure of parking event info 591](#_Toc149055527)

[4.27.414 Structure of vehicle entry image 592](#_Toc149055528)

[4.27.415 Structure of vehicle leaving image 593](#_Toc149055529)

[4.27.416 Structure of time template configuration 593](#_Toc149055530)

[4.27.417 Structure of exceptions of monitoring task schedule 594](#_Toc149055531)

[4.27.418 Structure of detailed info about daily arming schedule 594](#_Toc149055532)

[4.27.419 Structure of general-purpose ID list 595](#_Toc149055533)

[4.27.420 Structure of time template 595](#_Toc149055534)

[4.27.421 Structure of access control person info 596](#_Toc149055535)

[4.27.422 Structure of image info 597](#_Toc149055536)

[4.27.423 Structure of staff info 597](#_Toc149055537)

[4.27.424 Structure of time info 598](#_Toc149055538)

[4.27.425 Structure of visitor info 598](#_Toc149055539)

[4.27.426 Structure of info about person’s access control card 599](#_Toc149055540)

[4.27.427 Structure of blocklist info 599](#_Toc149055541)

[4.27.428 Structure of person list 600](#_Toc149055542)

[4.27.429 Structure of permission info 600](#_Toc149055543)

[4.27.430 Structure of batch operation list of face recognition module 601](#_Toc149055544)

[4.27.431 Structure of batch operation info of face recognition module 601](#_Toc149055545)

[4.27.432 Structure of entry/exit record info 601](#_Toc149055546)

[4.27.433 Structure of face comparison info 602](#_Toc149055547)

[4.27.434 Structure of search criteria 603](#_Toc149055548)

[4.27.435 Structure of basic info about access control persons 603](#_Toc149055549)

[4.27.436 Structure of visitor record info 604](#_Toc149055550)

[4.27.437 Structure of door permission info 605](#_Toc149055551)

[4.27.438 Structure of permission group info 605](#_Toc149055552)

[4.27.439 Structure of person verification 606](#_Toc149055553)

[4.27.440 Structure of face info 607](#_Toc149055554)

[4.27.441 Structure of card info 607](#_Toc149055555)

[4.27.442 Structure of gate info 608](#_Toc149055556)

[4.27.443 Structure of library comparison info 609](#_Toc149055557)

[4.27.444 Structure of matching person info 610](#_Toc149055558)

[4.27.445 Structure of image info 610](#_Toc149055559)

[4.27.446 Structure of plate monitoring and alarm info 611](#_Toc149055560)

[4.27.447 Vehicle member info 611](#_Toc149055561)

[4.27.448 Vehicle info 612](#_Toc149055562)

[4.27.449 Plate info 612](#_Toc149055563)

[4.27.450 Structure of vehicle list 612](#_Toc149055564)

[4.27.451 Structure of batch operation member list 613](#_Toc149055565)

[4.27.452 Structure of returned alarm records 613](#_Toc149055566)

[4.27.453 Structure of vehicle attribute info 614](#_Toc149055567)

[4.27.454 Structure of plate monitoring and alarm info 614](#_Toc149055568)

[4.27.455 Structure of transparent channel creation 615](#_Toc149055569)

[4.27.456 Structure of user info 615](#_Toc149055570)

[4.27.457 Structure of channel permission info 616](#_Toc149055571)

[4.27.458 Structure of modifying detailed user info 616](#_Toc149055572)

[4.27.459 Structure of user password info 617](#_Toc149055573)

[4.27.460 Structure of user list 617](#_Toc149055574)

[4.27.461 Structure of organization info 617](#_Toc149055575)

[4.27.462 Structure of info about organizations to be deleted 618](#_Toc149055576)

[4.27.463 Structure of response to organization deletion 618](#_Toc149055577)

[4.27.464 Structure of info about channels in an organization 619](#_Toc149055578)

[4.27.465 Structure of info about channels in an organization 619](#_Toc149055579)

[4.27.466 Structure of info about network port number and status 620](#_Toc149055580)

[4.27.467 Structure of protocol info 620](#_Toc149055581)

[4.27.468 Structure of monitoring info search criteria 621](#_Toc149055582)

[4.27.469 Structure of criteria for searching organization list 621](#_Toc149055583)

[4.27.470 Area people counting rule info 621](#_Toc149055584)

[4.27.471 Area detection rule info 622](#_Toc149055585)

[4.27.472 Area location info 622](#_Toc149055586)

[4.27.473 Coordinates of each vertex of the detection area 623](#_Toc149055587)

[4.27.474 Tripwire people counting rule info 623](#_Toc149055588)

[4.27.475 Tripwire detection rule info 623](#_Toc149055589)

[4.27.476 Tripwire people counting reset info 624](#_Toc149055590)

[4.27.477 List of linked dome cameras 624](#_Toc149055591)

[4.27.478 Info about linked dome camera 625](#_Toc149055592)

[4.27.479 System time configuration 625](#_Toc149055593)

[4.27.480 Structure of object list 626](#_Toc149055594)

[4.27.481 Structure of VIID info 626](#_Toc149055595)

[4.27.482 Structure of temperature info 627](#_Toc149055596)

[4.27.483 Recording time info 627](#_Toc149055597)

[4.27.484 Recording time list 628](#_Toc149055598)

[4.27.485 Detection area vertex coordinates info 628](#_Toc149055599)

[4.27.486 Rule info 628](#_Toc149055600)

[4.27.487 Info on a single object 629](#_Toc149055601)

[4.27.488 Audio output parameters 629](#_Toc149055602)

[4.27.489 Audio file information 630](#_Toc149055603)

[4.27.490 Channel list 630](#_Toc149055604)

[4.27.491 Structure of channel search criteria 630](#_Toc149055605)

[4.27.492 Image search criteria 631](#_Toc149055606)

[4.27.493 Image search results 631](#_Toc149055607)

[4.27.494 Object result info 632](#_Toc149055608)

[4.27.495 Object info list 632](#_Toc149055609)

[4.27.496 Alarm related data 633](#_Toc149055610)

[4.27.497 Basic info about alarm image 633](#_Toc149055611)

[4.27.498 Alarm snapshot data 634](#_Toc149055612)

[4.27.499 People counting alarm info 634](#_Toc149055613)

[4.27.500 Area people counting statistics info 635](#_Toc149055614)

[4.27.501 Tripwire people counting statistics info 635](#_Toc149055615)

[4.27.502 Crowd density statistics info 636](#_Toc149055616)

[4.27.503 Area people counting statistics 636](#_Toc149055617)

[4.27.504 Tripwire people counting statistics 637](#_Toc149055618)

[4.27.505 Crowd density counting statistics 637](#_Toc149055619)

[4.27.506 Crowd density statistics summary info 638](#_Toc149055620)

[4.27.507 Crowd density group info 638](#_Toc149055621)

[4.27.508 Channel rule info 639](#_Toc149055622)

[4.27.509 Crowd density rule info 639](#_Toc149055623)

[4.27.510 Alarm rule Info 640](#_Toc149055624)

[4.27.511 Info on a single object 640](#_Toc149055625)

[4.27.512 Fire detection alarm info 640](#_Toc149055626)

[4.27.513 Fire detection channel info 641](#_Toc149055627)

[4.27.514 Fire info list 641](#_Toc149055628)

[4.27.515 Fire position info 642](#_Toc149055629)

[4.27.516 Smoking Detection Info 642](#_Toc149055630)

[4.27.517 Smoking Detection Channel Info 643](#_Toc149055631)

[4.27.518 Temperature detection alarm info 643](#_Toc149055632)

[4.27.519 Temperature detection channel info 644](#_Toc149055633)

[4.27.520 Fire detection info 644](#_Toc149055634)

[4.27.521 Smoking info. 645](#_Toc149055635)

[4.27.522 Temperature info 645](#_Toc149055636)

[4.28 Definition of Enumerations 646](#_Toc149055637)

[4.28.1 Enumeration of access protocols 646](#_Toc149055638)

[4.28.2 Enumeration of exception message types 646](#_Toc149055639)

[4.28.3 Enumeration of device types 647](#_Toc149055640)

[4.28.4 Enumeration of stream types 647](#_Toc149055641)

[4.28.5 Enumeration of media transport protocols 648](#_Toc149055642)

[4.28.6 Enumeration of image fluency 648](#_Toc149055643)

[4.28.7 Enumeration of stream modes 648](#_Toc149055644)

[4.28.8 Enumeration of transport types 648](#_Toc149055645)

[4.28.9 Enumeration of streaming protocols 648](#_Toc149055646)

[4.28.10 Enumeration of media file formats 649](#_Toc149055647)

[4.28.11 Enumeration of snapshot image formats 650](#_Toc149055648)

[4.28.12 Enumeration of audio formats 650](#_Toc149055649)

[4.28.13 Enumeration of audio bit widths 650](#_Toc149055650)

[4.28.14 Enumeration of media stream formats 650](#_Toc149055651)

[4.28.15 Enumeration of audio encoding formats 651](#_Toc149055652)

[4.28.16 Enumeration of video frame types 651](#_Toc149055653)

[4.28.17 Enumeration of video compression formats 651](#_Toc149055654)

[4.28.18 Enumeration of metadata display types 651](#_Toc149055655)

[4.28.19 Enumeration of video display scales 652](#_Toc149055656)

[4.28.20 Enumeration of recording storage types 652](#_Toc149055657)

[4.28.21 Enumeration of recording storage locations 653](#_Toc149055658)

[4.28.22 Enumeration of recording search locations 653](#_Toc149055659)

[4.28.23 Enumeration of recording status 653](#_Toc149055660)

[4.28.24 Enumeration of recording download speeds 654](#_Toc149055661)

[4.28.25 Enumeration of play/download speeds 654](#_Toc149055662)

[4.28.26 Enumeration of PTZ commands 655](#_Toc149055663)

[4.28.27 PTZ auto guard configuration commands 656](#_Toc149055664)

[4.28.28 PTZ auto guard modes 656](#_Toc149055665)

[4.28.29 PTZ home position commands 657](#_Toc149055666)

[4.28.30 Focus status 657](#_Toc149055667)

[4.28.31 Enumeration of patrol statuses 657](#_Toc149055668)

[4.28.32 Enumeration of PTZ patrol operation 657](#_Toc149055669)

[4.28.33 Enumeration of PTZ preset operation commands 657](#_Toc149055670)

[4.28.34 Enumeration of PTZ patrol operation 658](#_Toc149055671)

[4.28.35 Enumeration of PTZ directions 658](#_Toc149055672)

[4.28.36 Enumeration of playback control commands 658](#_Toc149055673)

[4.28.37 Enumeration of main log types 659](#_Toc149055674)

[4.28.38 Enumeration of log sub types 662](#_Toc149055675)

[4.28.39 Enumeration of alarm types 668](#_Toc149055676)

[4.28.40 Enumeration of alarm log search criteria 669](#_Toc149055677)

[4.28.41 Enumeration of alarm types 671](#_Toc149055678)

[4.28.42 Enumeration of face search types 677](#_Toc149055679)

[4.28.43 Enumeration of organization types 677](#_Toc149055680)

[4.28.44 Enumeration of modeling statuses 677](#_Toc149055681)

[4.28.45 Enumeration of mask wearing statuses 678](#_Toc149055682)

[4.28.46 Enumeration of logic types of search criteria 678](#_Toc149055683)

[4.28.47 Enumeration of alarm sub types 678](#_Toc149055684)

[4.28.48 Enumeration of alarm source types 689](#_Toc149055685)

[4.28.49 Enumeration of member genders 689](#_Toc149055686)

[4.28.50 Enumeration of ID types 690](#_Toc149055687)

[4.28.51 Enumeration of file info 690](#_Toc149055688)

[4.28.52 Enumeration of face processing result codes 690](#_Toc149055689)

[4.28.53 Enumeration of person comparison results 691](#_Toc149055690)

[4.28.54 Enumeration of linkage actions 691](#_Toc149055691)

[4.28.55 Enumeration alarm output status 692](#_Toc149055692)

[4.28.56 Enumeration of arming schedule enablement 692](#_Toc149055693)

[4.28.57 Enumeration of days of a week 693](#_Toc149055694)

[4.28.58 Enumeration of arming types 693](#_Toc149055695)

[4.28.59 Enumeration of person library types 693](#_Toc149055696)

[4.28.60 Enumeration of face monitoring operation result codes 694](#_Toc149055697)

[4.28.61 Enumeration of alarm snapshot types 695](#_Toc149055698)

[4.28.62 Enumeration of 5ePTZ display modes 696](#_Toc149055699)

[4.28.63 Enumeration of fisheye dewarping modes 696](#_Toc149055700)

[4.28.64 Enumeration of device mounting modes 696](#_Toc149055701)

[4.28.65 Enumeration of mouse movement modes 697](#_Toc149055702)

[4.28.66 Enumeration of recording types 697](#_Toc149055703)

[4.28.67 Enumeration of recording status 697](#_Toc149055704)

[4.28.68 Device configuration commands 697](#_Toc149055705)

[4.28.69 IP protocol types 706](#_Toc149055706)

[4.28.70 Image quality 707](#_Toc149055707)

[4.28.71 Main stream types 707](#_Toc149055708)

[4.28.72 Bitrate types 707](#_Toc149055709)

[4.28.73 GOP types 707](#_Toc149055710)

[4.28.74 Extended image encoding modes 708](#_Toc149055711)

[4.28.75 Enumeration of OSD time formats 708](#_Toc149055712)

[4.28.76 OSD types 708](#_Toc149055713)

[4.28.77 Enumeration of OSD font styles 708](#_Toc149055714)

[4.28.78 Enumeration of OSD font sizes 708](#_Toc149055715)

[4.28.79 Date format 709](#_Toc149055716)

[4.28.80 Enumeration of OSD time formats 709](#_Toc149055717)

[4.28.81 Enumeration of OSD alignments 710](#_Toc149055718)

[4.28.82 Enumeration of OSD margin sizes 710](#_Toc149055719)

[4.28.83 Operation modes of alarm input 710](#_Toc149055720)

[4.28.84 Commands used to control manual alarm output 710](#_Toc149055721)

[4.28.85 Exposure modes 710](#_Toc149055722)

[4.28.86 Supported shutter times 711](#_Toc149055723)

[4.28.87 Metering control modes 712](#_Toc149055724)

[4.28.88 Day/night mode types 712](#_Toc149055725)

[4.28.89 Illuminator types 712](#_Toc149055726)

[4.28.90 White balance modes 712](#_Toc149055727)

[4.28.91 Enumeration of HDD working modes 713](#_Toc149055728)

[4.28.92 Focus modes 713](#_Toc149055729)

[4.28.93 Day/night mode 713](#_Toc149055730)

[4.28.94 Enumeration of defog modes 714](#_Toc149055731)

[4.28.95 Enumeration of audio collection ports 714](#_Toc149055732)

[4.28.96 Enumeration of audio encoding formats 714](#_Toc149055733)

[4.28.97 Enumeration of audio sampling rates 714](#_Toc149055734)

[4.28.98 Enumeration of audio input modes 715](#_Toc149055735)

[4.28.99 DST offset time 715](#_Toc149055736)

[4.28.100 Week 715](#_Toc149055737)

[4.28.101 Enumeration pre-alarm recording time 716](#_Toc149055738)

[4.28.102 Enumeration of post-alarm recording time 716](#_Toc149055739)

[4.28.103 Enumeration of motion detection area types 716](#_Toc149055740)

[4.28.104 HDD types 716](#_Toc149055741)

[4.28.105 HDD working modes 717](#_Toc149055742)

[4.28.106 HDD status 717](#_Toc149055743)

[4.28.107 Storage container status 717](#_Toc149055744)

[4.28.108 Storage container attributes 717](#_Toc149055745)

[4.28.109 Address types 718](#_Toc149055746)

[4.28.110 Storage container usage 718](#_Toc149055747)

[4.28.111 HDD health assessment status 718](#_Toc149055748)

[4.28.112 HDD S.M.A.R.T. test status 718](#_Toc149055749)

[4.28.113 HDD S.M.A.R.T. test types 719](#_Toc149055750)

[4.28.114 Access protocols of management server 719](#_Toc149055751)

[4.28.115 Storage space unit 719](#_Toc149055752)

[4.28.116 Transport protocol 719](#_Toc149055753)

[4.28.117 Storage resources 720](#_Toc149055754)

[4.28.118 Overwrite policy 720](#_Toc149055755)

[4.28.119 Live view index 720](#_Toc149055756)

[4.28.120 Image storage modes 720](#_Toc149055757)

[4.28.121 Back focus control commands 721](#_Toc149055758)

[4.28.122 Time zone 721](#_Toc149055759)

[4.28.123 Illuminator control modes 722](#_Toc149055760)

[4.28.124 HDD S.M.A.R.T. test result 722](#_Toc149055761)

[4.28.125 Communication protocol of photo server 723](#_Toc149055762)

[4.28.126 Sub communication protocols of photo server 723](#_Toc149055763)

[4.28.127 Access protocols of smart server 723](#_Toc149055764)

[4.28.128 OSD content type 724](#_Toc149055765)

[4.28.129 Enumeration of device types 724](#_Toc149055766)

[4.28.130 IP address types 725](#_Toc149055767)

[4.28.131 Lock signals 725](#_Toc149055768)

[4.28.132 Channel type 725](#_Toc149055769)

[4.28.133 Status of encoding channel 726](#_Toc149055770)

[4.28.134 Alarm input channel status 726](#_Toc149055771)

[4.28.135 Alarm output channel status 726](#_Toc149055772)

[4.28.136 Enumeration of video input types 727](#_Toc149055773)

[4.28.137 Enumeration of language types 727](#_Toc149055774)

[4.28.138 Enumeration of client types 728](#_Toc149055775)

[4.28.139 Enumeration of channel statuses 728](#_Toc149055776)

[4.28.140 Channel type 729](#_Toc149055777)

[4.28.141 Video standard 729](#_Toc149055778)

[4.28.142 Enumeration of PoE port status 729](#_Toc149055779)

[4.28.143 Enumeration of device offline causes 729](#_Toc149055780)

[4.28.144 Device status 730](#_Toc149055781)

[4.28.145 Device sub types 730](#_Toc149055782)

[4.28.146 Access protocol 731](#_Toc149055783)

[4.28.147 Access mode 731](#_Toc149055784)

[4.28.148 Types of devices connected to channels (for NVR only) 731](#_Toc149055785)

[4.28.149 Alarm/event report type 732](#_Toc149055786)

[4.28.150 Alarm resource types 732](#_Toc149055787)

[4.28.151 Enumerations of media types 732](#_Toc149055788)

[4.28.152 Enumeration of types of operation info reported by the decoding layer 733](#_Toc149055789)

[4.28.153 Enumeration of event action types 734](#_Toc149055790)

[4.28.154 Enumeration of event resources types 734](#_Toc149055791)

[4.28.155 People counting types 736](#_Toc149055792)

[4.28.156 People counting report types 736](#_Toc149055793)

[4.28.157 People counting report types 737](#_Toc149055794)

[4.28.158 Enumeration of resolutions 737](#_Toc149055795)

[4.28.159 Enumeration of window splitting capability 738](#_Toc149055796)

[4.28.160 Enumeration of window roaming capability 738](#_Toc149055797)

[4.28.161 Enumeration of window opening capability 738](#_Toc149055798)

[4.28.162 Enumeration of video output formats 738](#_Toc149055799)

[4.28.163 Enumeration of video port types 740](#_Toc149055800)

[4.28.164 Enumeration of audio channel ports 740](#_Toc149055801)

[4.28.165 Enumeration of serial ports 740](#_Toc149055802)

[4.28.166 Enumeration of virtual LED types 740](#_Toc149055803)

[4.28.167 Enumeration of optional command words for modifying virtual LEDs 741](#_Toc149055804)

[4.28.168 Enumeration of virtual LED font format 741](#_Toc149055805)

[4.28.169 Enumeration of virtual LED font sizes 741](#_Toc149055806)

[4.28.170 Enumeration of virtual LED font spacing 742](#_Toc149055807)

[4.28.171 Enumeration of horizontal text alignment modes 742](#_Toc149055808)

[4.28.172 Enumeration of vertical text alignment modes 742](#_Toc149055809)

[4.28.173 Enumeration of text scroll modes 742](#_Toc149055810)

[4.28.174 Enumeration of window magnification types 743](#_Toc149055811)

[4.28.175 Enumeration of window types 743](#_Toc149055812)

[4.28.176 Enumeration of zoom statuses 743](#_Toc149055813)

[4.28.177 Enumeration of video service in split window 743](#_Toc149055814)

[4.28.178 Enumeration of screen magnification types 744](#_Toc149055815)

[4.28.179 Enumeration of screen magnification modes 744](#_Toc149055816)

[4.28.180 Enumeration of media transport protocols 744](#_Toc149055817)

[4.28.181 Enumeration of live view types 744](#_Toc149055818)

[4.28.182 Enumeration of decoding types 744](#_Toc149055819)

[4.28.183 Enumeration of encryption types 745](#_Toc149055820)

[4.28.184 Enumeration of text positions 745](#_Toc149055821)

[4.28.185 Enumeration of resource management types 745](#_Toc149055822)

[4.28.186 Enumeration of stream types 745](#_Toc149055823)

[4.28.187 Enumeration of scene modification types 746](#_Toc149055824)

[4.28.188 Enumeration of scene modes 746](#_Toc149055825)

[4.28.189 Enumeration of background image types 746](#_Toc149055826)

[4.28.190 Enumeration of sequence statuses 746](#_Toc149055827)

[4.28.191 Enumeration of sequence playing control 747](#_Toc149055828)

[4.28.192 Enumeration of scene sequence plan modes 747](#_Toc149055829)

[4.28.193 Enumeration of encoding formats 747](#_Toc149055830)

[4.28.194 Enumeration of command words for modifying scene windows 747](#_Toc149055831)

[4.28.195 Enumeration of sequence statuses 748](#_Toc149055832)

[4.28.196 Enumeration of stream service types 748](#_Toc149055833)

[4.28.197 Enumeration of access control person management commands 748](#_Toc149055834)

[4.28.198 Enumeration of alarm point channel control commands 749](#_Toc149055835)

[4.28.199 Enumeration of manual linkage action types 749](#_Toc149055836)

[4.28.200 Enumeration of face member ID types 749](#_Toc149055837)

[4.28.201 Enumeration of image types 749](#_Toc149055838)

[4.28.202 Enumeration of image formats 750](#_Toc149055839)

[4.28.203 Enumeration of face pass-through record types 750](#_Toc149055840)

[4.28.204 Enumeration of age ranges 750](#_Toc149055841)

[4.28.205 Enumeration of glasses wearing status 750](#_Toc149055842)

[4.28.206 Enumeration of glasses styles 751](#_Toc149055843)

[4.28.207 Enumeration of sleeve lengths 751](#_Toc149055844)

[4.28.208 Enumeration of clothes colors 751](#_Toc149055845)

[4.28.209 Enumeration of trousers length 752](#_Toc149055846)

[4.28.210 Enumeration of captured body directions 752](#_Toc149055847)

[4.28.211 Enumeration of shoe tube lengths 752](#_Toc149055848)

[4.28.212 Enumeration of hair lengths 753](#_Toc149055849)

[4.28.213 Enumeration of bag carrying status 753](#_Toc149055850)

[4.28.214 Enumeration of mask wearing statuses 753](#_Toc149055851)

[4.28.215 Enumeration of coat textures 753](#_Toc149055852)

[4.28.216 Enumeration of person movement directions 754](#_Toc149055853)

[4.28.217 Enumeration of speed types of non-motor vehicles in structured scenes 754](#_Toc149055854)

[4.28.218 Enumeration of movement directions (relative to image) of non-motor vehicles in structured scenes 754](#_Toc149055855)

[4.28.219 Enumeration of non-motor vehicle types 755](#_Toc149055856)

[4.28.220 Enumeration of vehicle types 755](#_Toc149055857)

[4.28.221 Enumeration of plate colors 756](#_Toc149055858)

[4.28.222 Enumeration of plate types 756](#_Toc149055859)

[4.28.223 Enumeration of notification types 757](#_Toc149055860)

[4.28.224 Enumeration of smart alarm types 757](#_Toc149055861)

[4.28.225 Enumeration of LAPI subscription types 758](#_Toc149055862)

[4.28.226 Enumeration of lane types 758](#_Toc149055863)

[4.28.227 Enumeration of vehicle releasing types 758](#_Toc149055864)

[4.28.228 Enumeration of order payment statuses 758](#_Toc149055865)

[4.28.229 Enumeration of payment types 759](#_Toc149055866)

[4.28.230 Enumeration of person types 759](#_Toc149055867)

[4.28.231 Enumeration of visitor statuses 759](#_Toc149055868)

[4.28.232 Enumeration of collection sources 759](#_Toc149055869)

[4.28.233 Enumeration of match statuses 760](#_Toc149055870)

[4.28.234 Enumeration of image formats 760](#_Toc149055871)

[4.28.235 Plate colors 760](#_Toc149055872)

[4.28.236 Plate types 761](#_Toc149055873)

[4.28.237 Vehicle types 762](#_Toc149055874)

[4.28.238 Enumeration of speed types of non-motor vehicles in structured scenes 763](#_Toc149055875)

[4.28.239 Enumeration of movement directions (relative to image) of non-motor vehicles in structured scenes 763](#_Toc149055876)

[4.28.240 Enumeration of channel permissions 763](#_Toc149055877)

[4.28.241 Enumeration of response statuses 764](#_Toc149055878)

[4.28.242 Enumeration of NAT types 764](#_Toc149055879)

[4.28.243 Enumeration of protocol types 764](#_Toc149055880)

[4.28.244 Enumeration of organization search modes 764](#_Toc149055881)

[4.28.245 Enumeration of basic permissions 765](#_Toc149055882)

[4.28.246 HDD modes 765](#_Toc149055883)

[4.28.247 Capability commands 765](#_Toc149055884)

[4.28.248 OSD date format capability 766](#_Toc149055885)

[4.28.249 Enumeration of time sync modes 766](#_Toc149055886)

[4.28.250 Stream data callback function 766](#_Toc149055887)

[4.28.251 People counting types 767](#_Toc149055888)

[4.28.252 Enumeration of listening types 767](#_Toc149055889)

[4.28.253 Enumeration of supported alarm types 767](#_Toc149055890)

[4.28.254 Enumeration of plate colors 768](#_Toc149055891)

[4.28.255 Enumeration of recording search types 768](#_Toc149055892)

[4.28.256 Enumeration of skin tones 769](#_Toc149055893)

[4.28.257 Emotions 769](#_Toc149055894)

[4.28.258 Smile flag 770](#_Toc149055895)

[4.28.259 Beard flag 770](#_Toc149055896)

[4.28.260 Enumeration of object types 770](#_Toc149055897)

[4.28.261 Enumeration of rule types 770](#_Toc149055898)

[4.28.262 Enumeration of rule trigger types 771](#_Toc149055899)

[4.28.263 Target record types 771](#_Toc149055900)

[4.28.264 People counting alarm types 771](#_Toc149055901)

[4.28.265 Crowd density alarm statistics type 771](#_Toc149055902)

[4.28.266 Temperature alarm value types 772](#_Toc149055903)

[4.29 Error Code List 772](#_Toc149055904)

[4.29.1 Common Error Codes 772](#_Toc149055905)

[4.29.2 Stream Media Related Error Code 774](#_Toc149055906)

[4.29.3 Error Codes of PTZ Module 778](#_Toc149055907)

[4.29.4 Error Code of Playback Module 779](#_Toc149055908)

[4.29.5 Error Code of Storage Module 779](#_Toc149055909)

[4.29.6 Error Code of User Management Module 780](#_Toc149055910)

[4.29.7 Error Code of Device Management Module 780](#_Toc149055911)

[4.29.8 Error Code of Organization Management Module 781](#_Toc149055912)

[4.29.9 Error Code of Video Wall Management Module 782](#_Toc149055913)

[4.29.10 Network Related Error Code 782](#_Toc149055914)

[5. Definition of Product Related Interfaces 784](#_Toc149055915)

[5.1 Common Interfaces 784](#_Toc149055916)

[5.2 Network Camera 787](#_Toc149055917)

[5.3 NVR 790](#_Toc149055918)

[5.4 VMS 793](#_Toc149055919)

[6. Help 798](#_Toc149055920)

[6.1 Programming Environment Setup 798](#_Toc149055921)

[6.1.1 Development and Compilation Environment on Windows 798](#_Toc149055922)

[6.1.2 Development and Compilation Environment on Linux 799](#_Toc149055923)

[6.1.3 Development and Compilation Environment on Android 799](#_Toc149055924)

[6.1.4 Development and Compilation Environment on iOS 800](#_Toc149055925)

[6.2 Precautions 801](#_Toc149055926)

[6.3 FAQ 801](#_Toc149055927)

# Introduction

## Disclaimer

Thank you for choosing our product. Should you have any questions, please do not hesitate to contact us.

This manual is only for informational purpose, and all statements, information, and recommendations in this manual are presented without warranty of any kind, expressed or implied.

Users must assume total responsibility and all risks for using this manual or our products. In no event shall we be liable for any special, incidental, indirect, or consequential damages, including, without limitation, lost profits or loss or damage to data arising out of the use or inability to use this manual or our products, or arising from abnormal operation of our product or data leakage due to network attack, hacker attack, virus infection, etc.

This manual may contain technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in the new editions of this manual. We may make improvements and/or changes in the product(s) and/or the program(s) described in this manual at any time without prior notice.

## Intended Audience

This document is intended for the development, maintenance or management personnel who have certain development capabilities, understand C/C++, C# and other languages, and have certain experience and understanding of SDK integration.

## Overview

NetDEVSDK (also known as network device SDK) is a set of supporting modules developed based on private network communication protocols to serve IPC, NVR, VMS and other products. It includes interfaces for surveillance management and control, device management, resource management, and system maintenance. Users can use these interfaces to made custom software development to achieve remote access and device control.

The SDK mainly provides video service functions such as device management, live view, playback, alarm, PTZ, and parameter configuration.

This manual details interface functions, the usage, and relationship between interfaces. It supports C/C++, C#, Objective-C, etc.

When you get this manual, you should have got the complete SDK, including header files, static libraries, dynamic libraries, demo, and developer documentation.

## Glossary

|  |  |
| --- | --- |
| Term | Description |
| Live view | View the live video of a camera |
| Playback | Search and play stored videos, control the playback. |
| Stream callback | During live view or playback, the SDK acts as the server and can obtain video stream data by setting the stream callback function. |
| Recording download | Download videos from a storage device to the local |
| Local recording | Record live or recorded video images being played and save to a local path. |
| Local snapshot | Take snapshots of live video images or video images being played back and save to a local path. |

## System Requirements

* **SDK for 32-bit Windows**

Windows 7, Windows 8, Windows 10

* **SDK for 64-bit Windows**

Windows 7, Windows 8, Windows 10

* **SDK for 64-bit Linux**

GCC 4.4.7 or later

OS: CentOS 6.5 or later

* **SDK for 32-bit Linux**

GCC 4.4.7 or later

OS: CentOS 6.5 or later

* **SDK for Android**

Supported system: Android 5 or later

* **SDK for iOS**

Supported system: iOS 8.0 or later

# Version Update

## Release Notes

|  |  |  |
| --- | --- | --- |
| **New Version** | **Revision Date** | **Description** |
| 2.5.15.0 | 2021-07-06 | Bug fixes |
| 2.6.0.0 | 2021-11-20 | 1. Bug fixes. 2. Added support for people counting. 3. Added support for fire detection alarm. 4. Added contents about customer needs. 5. Refined stream error codes. |
| 2.7.0.0 | 2023-06-28 | 1. Bug fixes. 2. Added support for smart intrusion prevention (SIP). 3. Added support for fire detection and thermal imaging. 4. Added support for high-rise littering and elevator entrance detection. |

# Programming Guide

## Overview

The SDK is used to access service function interfaces provided by IPC, NVR, and VMS. Users can develop various media services (live view, playback and recording download, PTZ control, and parameter configuration, etc) by following the basic framework of SDK programming and by referring to the interface descriptions provided in this guide.

* **SDK function interface files**

NetDEVSDK.h: Provides definitions of macros, enumerations, data structures, and functional interfaces for the SDK.

* **Note**

1. Character string format

Non-English characters must be UTF-8 encoded.

2. Time description

Unix time: Seconds that have elapsed since 1970 January 1 00:00:00 GMT

3. Error codes

False will be returned if a function encountered an exception inside after being called. Call [NETDEV\_GetLastError()](#_获取接口错误码_1) to get the error code. See Error Code List for details.

## Major Process

### Flowchart



The dotted lines indicate optional parts of the process, which has no impact on other processes or modules. As shown in this figure, the whole process is divided into different modules according to their functions. To implement the functions of each module, five processes are required: Initialization, Login, Get Channel Info, Logout, and Cleanup.

* Set log path [NETDEV\_SetLogPath](#_设置日志文件路径) is optional.
* Initialization [NETDEV\_Init](#_SDK_初始化): Initialize the whole SDK and complete operations such as memory preallocation.
* Device login: Applicable to devices on the local area network (LAN) and the public network, and the prerequisite is that the device is accessible from a local PC or a mobile phone browser. Local device login [NETDEV\_Login\_V30](#_用户设备登录): Implements user login; and after login succeeded, returns the user ID as the only identifier for other operations.
* Get IPC/NVR channel info: Use [NETDEV\_QueryVideoChlDetailListEx](#_查询视频通道信息列表) to get channel information after login succeeded.
* Get information about devices and channels under VMS: The following interfaces are used to get the list of devices under the VMS: [NETDEV\_FindDevList](#_通过设备类型查询设备列表), [NETDEV\_FindNextDevInfo](#_逐个获取查找到的设备信息), [NETDEV\_FindCloseDevInfo](#_关闭查找设备信息，释放资源). The following interfaces are used to get channel information by device ID. [NETDEV\_FindDevChnList](#_通过设备ID或通道类型查询通道信息列表), [NETDEV\_FindNextDevChn](#_逐个获取查找到的设备通道信息), [NETDEV\_FindCloseDevChn](#_关闭查找设备通道信息，释放资源).
* Exception module: Used to receive information about exceptions that occurred in live view, playback, download, and other modules. See Process of Exception Module.
* Alarm module: Mainly used to get real-time and historical alarms of devices. See [Process of Alarm Module](mk:@MSITStore:C:\Users\y03362\Downloads\NETDEVSDK_Win64_V2.5.0.9\doc\网络设备SDK开发手册_V2.5.0.chm::/编程指导/告警模块流程.htm).
* Live view module: Get live video streams from front-end devices and decode to display. See [Process of Live View Module](mk:@MSITStore:C:\Users\y03362\Downloads\NETDEVSDK_Win64_V2.5.0.9\doc\网络设备SDK开发手册_V2.5.0.chm::/编程指导/实况模块流程.htm).
* Playback & download module: Searches recorded videos and replays remotely, or downloads recording files from devices, and then decodes or saves to a storage device. See Process of Playback & Recording Download Module.
* Parameter configuration module: Get and set device parameters, mainly including video encoding parameters. See Process of Parameter Configuration Module.
* Two-way audio module: Two-way audio and broadcast operations. See Process of Audio Module.
* Maintenance module: Restore factory default settings, restart device, etc. See Process of Maintenance Module.
* PTZ module: Control pan/tilt, presets, patrol, etc. See Process of PTZ Module.
* Smart module: Searching faces, plates, people counting, etc. See Process of Smart Module.
* User Logout [NETDEV\_Logout](#_用户注销): User exits the system.
* Release SDK Resources [NETDEV\_Cleanup](#_SDK清理): Call this interface to release SDK resources before exiting the system.

### Sample code for NVR or IPC login

|  |
| --- |
| The following C++ sample code shows the device login process of NVR or IPC. For reference only. |
| #include <iostream>  #include "NetDEVSDK.h"  void main**()**  **{**    NETDEV\_Init**();**    NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.122"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "123456"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  NETDEV\_Cleanup**();**  **return;**  **}**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  **if** **(**TRUE **==** bRet**)**  **{**  printf**(**"QueryVideoChlDetailList Success\n"**);**  **}**  **else**  **{**  printf**(**"QueryVideoChlDetailList failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  **}**      NETDEV\_Logout**(**lUserID**);**    NETDEV\_Cleanup**();**  **}** |

### Sample code for VMS login

|  |
| --- |
| The following C++ sample code shows the device login process of VMS. For reference only. |
| #include <iostream>  #include "NetDEVSDK.h"  void main**()**  **{**    NETDEV\_Init**();**    NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.108"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "123456"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_PRIVATE**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**    NETDEV\_Cleanup**();**  **return;**  **}**  INT32 nDeviceType**[]** **=** **{**NETDEV\_DTYPE\_MAIN\_ENCODE**,** NETDEV\_DTYPE\_MAIN\_BAYONET**};**  **for** **(**INT32 i **=**0**;** i**<sizeof(**nDeviceType**)/sizeof(**INT32**);** i**++)**  **{**    LPVOID lpDevFindHandle **=** NETDEV\_FindDevList**(**lUserID**,** nDeviceType**[**i**]);**  **if(NULL** **==** lpDevFindHandle**)**  **{**  **continue;**  **}**    NETDEV\_DEV\_BASIC\_INFO\_S stDevInfo **=** **{**0**};**  **while(**TRUE **==** NETDEV\_FindNextDevInfo**(**lpDevFindHandle**,** **&**stDevInfo**))**  **{**    LPVOID lpChnFindHandle **=** NETDEV\_FindDevChnList**(**lUserID**,** stDevInfo**.**dwDevID**,** NETDEV\_CHN\_TYPE\_ENCODE**);**  **if(NULL** **==** lpChnFindHandle**)**  **{**  **continue;**  **}**  **else**  **{**    NETDEV\_DEV\_CHN\_ENCODE\_INFO\_S stDevChnEncodeInfo **=** **{**0**};**  INT32 dwBytesReturned**;**  **while(**TRUE **==** NETDEV\_FindNextDevChn**(**lpChnFindHandle**,** **&**stDevChnEncodeInfo**,sizeof(**NETDEV\_DEV\_CHN\_ENCODE\_INFO\_S**),&**dwBytesReturned**))**  **{**  printf**(**"DeviceType[%d],%s:ChannelID[%d]\n"**,** nDeviceType**[**i**],**stDevChnEncodeInfo**.**stChnBaseInfo**.**szChnName**,** stDevChnEncodeInfo**.**stChnBaseInfo**.**dwChannelID**);**  **}**    NETDEV\_FindCloseDevChn**(**lpChnFindHandle**);**  **}**  **}**  NETDEV\_FindCloseDevInfo**(**lpDevFindHandle**);**  **}**      NETDEV\_Logout**(**lUserID**);**    NETDEV\_Cleanup**();**  **}** |

## Process of Parameter Configuration Module

### Flowchart



### Sample code for image configuration

|  |
| --- |
| The following C++ sample code shows the image configuration process. For reference only. |
| #include <iostream>  #include "NetDEVSDK.h"  void main**()**  **{**    NETDEV\_Init**();**    NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.153"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "123456"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  NETDEV\_Cleanup**();**  **return;**  **}**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  INT32 dwBytesReturned **=** 0**;**  NETDEV\_IMAGE\_SETTING\_S stImageInfo **=** **{**0**};**  bRet **=** NETDEV\_GetDevConfig**(**lUserID**,** 1**,** NETDEV\_GET\_IMAGECFG**,** **&**stImageInfo**,** **sizeof(**NETDEV\_IMAGE\_SETTING\_S**),** **&**dwBytesReturned**);**  **if** **(**TRUE **!=** bRet**)**  **{**  printf**(**"NETDEV\_GetDevConfig NETDEV\_GET\_IMAGECFG error"**);**  **}**  **else**  **{**  stImageInfo**.**dwContrast **=** 100**;**  bRet **=** NETDEV\_SetDevConfig**(**lUserID**,** 1**,** NETDEV\_SET\_IMAGECFG**,** **&**stImageInfo**,** **sizeof(**NETDEV\_IMAGE\_SETTING\_S**));**  **if** **(**TRUE **!=** bRet**)**  **{**  printf**(**"NETDEV\_SetDevConfig NETDEV\_SET\_IMAGECFG error"**);**  **}**  **else**  **{**  NETDEV\_GetDevConfig**(**lUserID**,** 1**,** NETDEV\_GET\_IMAGECFG**,** **&**stImageInfo**,** **sizeof(**NETDEV\_IMAGE\_SETTING\_S**),** **&**dwBytesReturned**);**  printf**(**"Image Contrast is:%d\n"**,** stImageInfo**.**dwContrast**);**  **}**  **}**  NETDEV\_Logout**(**lUserID**);**  NETDEV\_Cleanup**();**  **}** |

## Process of Playback & Recording Download Module

### Flowchart



### Sample code for recording search and playback by time

|  |
| --- |
| The following C++ sample code shows the process of recording search and playback by time. For reference only. |
| #include <iostream>  #include <time.h>  #include "Windows.h"  #include "NetDEVSDK.h"  void playBack**(**LPVOID lpUserID**,**INT32 dwChannelID**,** NETDEV\_FINDDATA\_S stVodFile**)**  **{**  HWND hWnd **=** GetConsoleWindow**();**  NETDEV\_PLAYBACKCOND\_S stPlayBackInfo **=** **{**0**};**  stPlayBackInfo**.**dwChannelID **=** dwChannelID**;**  stPlayBackInfo**.**tBeginTime **=** stVodFile**.**tBeginTime**;**  stPlayBackInfo**.**tEndTime **=** stVodFile**.**tEndTime**;**  stPlayBackInfo**.**hPlayWnd **=** hWnd**;**  stPlayBackInfo**.**dwLinkMode **=** 1**;**  LPVOID lpPlayHandle **=** NETDEV\_PlayBackByTime**(**lpUserID**,** **&**stPlayBackInfo**);**  **if** **(NULL** **==** lpPlayHandle**)**  **{**  printf**(** "PlayBackByTime failed. error[%d]\n"**,** NETDEV\_GetLastError**());**  **return;**  **}**  Sleep**(**1000**);**  INT64 iPlayTime **=** 0**;**  BOOL bRet **=** NETDEV\_PlayBackControl**(**lpPlayHandle**,**NETDEV\_PLAY\_CTRL\_GETPLAYTIME**,** **&**iPlayTime**);**  **for(**INT64 tTime **=** stVodFile**.**tBeginTime**;** iPlayTime **<** stVodFile**.**tEndTime**;** **)**  **{**  NETDEV\_PlayBackControl**(**lpPlayHandle**,**NETDEV\_PLAY\_CTRL\_GETPLAYTIME**,** **&**iPlayTime**);**    Sleep**(**1000**);**  **}**  NETDEV\_StopPlayBack**(**lpPlayHandle**);**  **}**  void main**()**  **{**    NETDEV\_Init**();**    NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**    NETDEV\_Cleanup**();**  **return;**  **}**    NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**    NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**    **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**    NETDEV\_FILECOND\_S stFileCond **=** **{**0**};**  stFileCond**.**dwChannelID **=** 1**;**  time\_t tUTCTime **=** time**(NULL);**  stFileCond**.**tEndTime **=** time**(&**tUTCTime**);**  stFileCond**.**tBeginTime **=** time**(&**tUTCTime**)-**60**;**  LPVOID dwFileHandle **=** NETDEV\_FindFile**(**lUserID**,** **&**stFileCond**);**  **if(NULL** **==** dwFileHandle**)**  **{**  printf**(**"FindFile fail, error code: %d\n"**,** NETDEV\_GetLastError**());**  **}**  **else**  **{**    NETDEV\_FINDDATA\_S stVodFile **=** **{**0**};**  **while** **(**NETDEV\_FindNextFile**(**dwFileHandle**,** **&**stVodFile**))**  **{**  printf**(**"FileName:%s, time:%lld——%lld\n"**,** stVodFile**.**szFileName**,** stVodFile**.**tBeginTime**,** stVodFile**.**tEndTime**);**    playBack**(**lUserID**,** stFileCond**.**dwChannelID**,** stVodFile**);**  **}**    NETDEV\_FindClose**(**dwFileHandle**);**  **}**    NETDEV\_Logout**(**lUserID**);**    NETDEV\_Cleanup**();**  **}** |

### Sample code for recording search and download by time

|  |
| --- |
| The following C++ sample code shows the process of recording search and download by time. For reference only. |
| #include <iostream>  #include <time.h>  #include "Windows.h"  #include "NetDEVSDK.h"  void saveRecordFile**(**LPVOID lpUserID**,**NETDEV\_FINDDATA\_S stVodFile**)**  **{**    NETDEV\_PLAYBACKCOND\_S stPlayBackInfo **=** **{**0**};**  stPlayBackInfo**.**dwChannelID **=** 1**;**  stPlayBackInfo**.**tBeginTime **=** stVodFile**.**tBeginTime**;**  stPlayBackInfo**.**tEndTime **=** stVodFile**.**tEndTime**;**  stPlayBackInfo**.**dwDownloadSpeed **=** NETDEV\_DOWNLOAD\_SPEED\_EIGHT**;**  LPVOID lpPlayHandle **=** NETDEV\_GetFileByTime**(**lpUserID**,** **&**stPlayBackInfo**,** "./test"**,** 0**);**  **if** **(NULL** **==** lpPlayHandle**)**  **{**  printf**(** "GetFileByTime failed. error[%d]\n"**,** NETDEV\_GetLastError**());**  **return;**  **}**  Sleep**(**1000**);**  INT64 iPlayTime **=** 0**;**  BOOL bRet **=** NETDEV\_PlayBackControl**(**lpPlayHandle**,**NETDEV\_PLAY\_CTRL\_GETPLAYTIME**,** **&**iPlayTime**);**  **for(**INT64 tTime **=** stVodFile**.**tBeginTime**;** iPlayTime **<** stVodFile**.**tEndTime**;** **)**  **{**  NETDEV\_PlayBackControl**(**lpPlayHandle**,**NETDEV\_PLAY\_CTRL\_GETPLAYTIME**,** **&**iPlayTime**);**  printf**(**"FileName:%s, time:%lld——%lld, iPlayTime=%lld\n"**,** stVodFile**.**szFileName**,** stVodFile**.**tBeginTime**,** stVodFile**.**tEndTime**,** iPlayTime**);**  Sleep**(**1000**);**  **}**    NETDEV\_StopGetFile**(**lpPlayHandle**);**  **}**  void main**()**  **{**    NETDEV\_Init**();**    NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**    NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**    NETDEV\_Cleanup**();**  **return;**  **}**    NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  NETDEV\_FILECOND\_S stFileCond **=** **{**0**};**  stFileCond**.**dwChannelID **=** 1**;**  time\_t tUTCTime **=** time**(NULL);**  stFileCond**.**tEndTime **=** time**(&**tUTCTime**);**  stFileCond**.**tBeginTime **=** time**(&**tUTCTime**)-**60**;**  LPVOID dwFileHandle **=** NETDEV\_FindFile**(**lUserID**,** **&**stFileCond**);**  **if(NULL** **==** dwFileHandle**)**  **{**  printf**(**"FindFile fail, error code: %d\n"**,** NETDEV\_GetLastError**());**  **}**  **else**  **{**  NETDEV\_FINDDATA\_S stVodFile **=** **{**0**};**  **while** **(**NETDEV\_FindNextFile**(**dwFileHandle**,** **&**stVodFile**))**  **{**  printf**(**"FileName:%s, time:%lld——%lld\n"**,** stVodFile**.**szFileName**,** stVodFile**.**tBeginTime**,** stVodFile**.**tEndTime**);**    saveRecordFile**(**lUserID**,** stVodFile**);**  **}**  NETDEV\_FindClose**(**dwFileHandle**);**  **}**  NETDEV\_Logout**(**lUserID**);**  NETDEV\_Cleanup**();**  **}** |

### Sample code for recording search and download by filename

|  |
| --- |
| The following C++ sample code shows the process of recording search and download by filename. For reference only. |
| #include <iostream>  #include <time.h>  #include "Windows.h"  #include "NetDEVSDK.h"  void saveRecordFile**(**LPVOID lpUserID**,**NETDEV\_FINDDATA\_S stVodFile**)**  **{**  NETDEV\_PLAYBACKINFO\_S stPlayBackInfo **=** **{**0**};**  stPlayBackInfo**.**tBeginTime **=** stVodFile**.**tBeginTime**;**  stPlayBackInfo**.**tEndTime **=** stVodFile**.**tEndTime**;**  stPlayBackInfo**.**dwDownloadSpeed **=** NETDEV\_DOWNLOAD\_SPEED\_EIGHT**;**  strcpy**(**stPlayBackInfo**.**szName**,** stVodFile**.**szFileName**);**  LPVOID lpPlayHandle **=** NETDEV\_GetFileByName**(**lpUserID**,** **&**stPlayBackInfo**,** "./test"**,** 0**);**  **if** **(NULL** **==** lpPlayHandle**)**  **{**  printf**(** "GetFileByName failed. error[%d]\n"**,** NETDEV\_GetLastError**());**  **return;**  **}**  Sleep**(**1000**);**  INT64 iPlayTime **=** 0**;**  BOOL bRet **=** NETDEV\_PlayBackControl**(**lpPlayHandle**,**NETDEV\_PLAY\_CTRL\_GETPLAYTIME**,** **&**iPlayTime**);**  **for(**INT64 tTime **=** stVodFile**.**tBeginTime**;** iPlayTime **<** stVodFile**.**tEndTime**;** **)**  **{**  NETDEV\_PlayBackControl**(**lpPlayHandle**,**NETDEV\_PLAY\_CTRL\_GETPLAYTIME**,** **&**iPlayTime**);**  printf**(**"FileName:%s, time:%lld——%lld, iPlayTime=%lld\n"**,** stVodFile**.**szFileName**,** stVodFile**.**tBeginTime**,** stVodFile**.**tEndTime**,** iPlayTime**);**  Sleep**(**1000**);**  **}**  NETDEV\_StopGetFile**(**lpPlayHandle**);**  **}**  void main**()**  **{**  NETDEV\_Init**();**  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**    NETDEV\_Cleanup**();**  **return;**  **}**    NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**    **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**    NETDEV\_FILECOND\_S stFileCond **=** **{**0**};**  stFileCond**.**dwChannelID **=** 1**;**  time\_t tUTCTime **=** time**(NULL);**  stFileCond**.**tEndTime **=** time**(&**tUTCTime**);**  stFileCond**.**tBeginTime **=** time**(&**tUTCTime**)-**60**;**  LPVOID dwFileHandle **=** NETDEV\_FindFile**(**lUserID**,** **&**stFileCond**);**  **if(NULL** **==** dwFileHandle**)**  **{**  printf**(**"FindFile fail, error code: %d\n"**,** NETDEV\_GetLastError**());**  **}**  **else**  **{**  NETDEV\_FINDDATA\_S stVodFile **=** **{**0**};**  **while** **(**NETDEV\_FindNextFile**(**dwFileHandle**,** **&**stVodFile**))**  **{**  printf**(**"FileName:%s, time:%lld——%lld\n"**,** stVodFile**.**szFileName**,** stVodFile**.**tBeginTime**,** stVodFile**.**tEndTime**);**    saveRecordFile**(**lUserID**,** stVodFile**);**  **}**    NETDEV\_FindClose**(**dwFileHandle**);**  **}**    NETDEV\_Logout**(**lUserID**);**    NETDEV\_Cleanup**();**  **}** |

## Process of Live View Module

### Flowchart



### Sample code for live view

|  |
| --- |
| The following C++ sample code shows the live view process. For reference only. |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void main**()**  **{**  NETDEV\_Init**();**  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  NETDEV\_Cleanup**();**  **return;**  **}**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**    HWND hWnd **=** GetConsoleWindow**();**  NETDEV\_PREVIEWINFO\_S stNetInfo **=** **{**0**};**  stNetInfo**.**dwChannelID **=** 1**;**  stNetInfo**.**hPlayWnd **=** hWnd**;**  stNetInfo**.**dwStreamType **=** 0**;**  stNetInfo**.**dwLinkMode **=** 1**;**  LPVOID lpPlayHandle **=** NETDEV\_RealPlay**(**lUserID**,** **&**stNetInfo**,** **NULL,** 0**);**  **if** **(NULL** **==** lpPlayHandle**)**  **{**  printf**(**"RealPlay failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  **}**  **else**  **{**  Sleep**(**20000**);**  NETDEV\_StopRealPlay**(**lpPlayHandle**);**  **}**    NETDEV\_Logout**(**lUserID**);**  NETDEV\_Cleanup**();**  **}** |

## Process of Audio Module

### Process of two-way audio module

#### Flowchart



#### Sample code

|  |
| --- |
| The following C++ sample code shows the two-way audio process. For reference only. |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void STDCALL NETDEV\_DECODE\_AUDIO\_DATA\_CALLBACK(LPVOID lpPlayHandle,const NETDEV\_WAVE\_DATA\_S \*pstWaveData,LPVOID lpUserParam)  {  printf("DataLen: %d\n", pstWaveData->dwDataLen);  }  void main()  {    NETDEV\_Init();    NETDEV\_REV\_TIMEOUT\_S stRevTimeout = {0};  stRevTimeout.dwRevTimeOut = 5;  stRevTimeout.dwFileReportTimeOut = 30;  NETDEV\_SetRevTimeOut(&stRevTimeout);    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo = {0};  strncpy(stDevLoginInfo.szIPAddr, "192.168.3.220", sizeof(stDevLoginInfo.szIPAddr));  strncpy(stDevLoginInfo.szUserName, "admin", sizeof(stDevLoginInfo.szUserName));  strncpy(stDevLoginInfo.szPassword, "Admin12345", sizeof(stDevLoginInfo.szPassword));  stDevLoginInfo.dwPort = 80;  stDevLoginInfo.dwLoginProto = NETDEV\_LOGIN\_PROTO\_ONVIF;    NETDEV\_SELOG\_INFO\_S stSELogInfo = {0};  LPVOID lUserID = NETDEV\_Login\_V30(&stDevLoginInfo, &stSELogInfo);  if (NULL == lUserID)  {  printf("Login failed, error code: %d\n", NETDEV\_GetLastError());    NETDEV\_Cleanup();  return;  }    NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx[128] = {0};  INT32 dwCount = 128;  BOOL bRet = NETDEV\_QueryVideoChlDetailListEx(lUserID, &dwCount, stVideoChlDetailInfoEx);  if (FALSE == bRet && NETDEV\_E\_NEEDMOREDATA == NETDEV\_GetLastError())  {  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S\* pstVideoChlDetailInfoEx = new NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S[dwCount];  memset(pstVideoChlDetailInfoEx, 0, sizeof(NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S)\*dwCount);  bRet = NETDEV\_QueryVideoChlDetailListEx(lUserID, &dwCount, pstVideoChlDetailInfoEx);  delete []pstVideoChlDetailInfoEx;  }  LPVOID lpPlayHandle = NETDEV\_StartVoiceCom(lUserID, 1, NETDEV\_DECODE\_AUDIO\_DATA\_CALLBACK, 0);  if (NULL == lpPlayHandle)  {  printf("StartVoiceCom failed, error code: %d\n", NETDEV\_GetLastError());  }  else  {  Sleep(20000);  NETDEV\_StopVoiceCom(lpPlayHandle);  }    NETDEV\_Logout(lUserID);  NETDEV\_Cleanup();  } |

### Process of audio forwarding module

#### Flowchart



#### Sample code

|  |
| --- |
| The following C++ sample code shows the audio forwarding process. For reference only. |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void STDCALL NETDEV\_DECODE\_AUDIO\_DATA\_CALLBACK**(**LPVOID lpPlayHandle**,**const NETDEV\_WAVE\_DATA\_S **\***pstWaveData**,**LPVOID lpUserParam**)**  **{**  printf**(**"DataLen: %d\n"**,** pstWaveData**->**dwDataLen**);**  **}**  void main**()**  **{**    NETDEV\_Init**();**    NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**    NETDEV\_Cleanup**();**  **return;**  **}**    NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**    **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  LPVOID lpPlayHandle **=** NETDEV\_StartInputVoiceSrv**(**lUserID**,** 1**);**  **if** **(NULL** **==** lpPlayHandle**)**  **{**  printf**(**"StartVoiceCom failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  **}**  **else**  **{**  NETDEV\_SetPlayDecodeAudioCB**(**lpPlayHandle**,** NETDEV\_DECODE\_AUDIO\_DATA\_CALLBACK**,** 1**,** **NULL);**    char szSendBuf**[**640**]** **=** **{**0**};**  int dwBufLen **=** 640**;**  NETDEV\_AUDIO\_SAMPLE\_PARAM\_S stVoiceParam **=** **{**0**};**  stVoiceParam**.**dwChannels **=** 1**;**  stVoiceParam**.**dwSampleRate **=** 16000**;**  stVoiceParam**.**enSampleFormat **=** NETDEV\_AUDIO\_SAMPLE\_FMT\_S16**;**  NETDEV\_InputVoiceData**(**lpPlayHandle**,** szSendBuf**,** dwBufLen**,** **&**stVoiceParam**);**  Sleep**(**20000**);**  NETDEV\_StopInputVoiceSrv**(**lpPlayHandle**);**  **}**    NETDEV\_Logout**(**lUserID**);**  NETDEV\_Cleanup**();**  **}** |

## Process of Maintenance Module

### Flowchart



### Sample code

|  |
| --- |
| The following C++ sample code shows the audio forwarding process. For reference only. |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void STDCALL NETDEV\_DECODE\_AUDIO\_DATA\_CALLBACK**(**LPVOID lpPlayHandle**,**const NETDEV\_WAVE\_DATA\_S **\***pstWaveData**,**LPVOID lpUserParam**)**  **{**  printf**(**"DataLen: %d\n"**,** pstWaveData**->**dwDataLen**);**  **}**  void main**()**  **{**    NETDEV\_Init**();**    NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  NETDEV\_Cleanup**();**  **return;**  **}**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  bRet **=** NETDEV\_Reboot**(**lUserID**);**  **if** **(**TRUE **!=** bRet**)**  **{**  printf**(**"Reboot failed\n"**);**  **}**    NETDEV\_Logout**(**lUserID**);**    NETDEV\_Cleanup**();**  **}** |

## Process of Exception Module

### Flowchart



### Sample code

|  |
| --- |
| The following C++ sample code shows the exception handling process. For reference only. |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void STDCALL ExceptionCallBack\_PF**(**IN LPVOID dwUserID**,**  IN INT32 dwType**,**  IN LPVOID dwHandle**,**  IN LPVOID dwUserData**,**  OUT LPNETDEV\_EXCEPTION\_OUTPUT\_INFO\_S pstExceptionOutputInfo  **)**  **{**  **switch(**dwType**)**  **{**  **case** NETDEV\_EXCEPTION\_EXCHANGE**:**  printf**(**"----------Device offline--------\n"**);**  **break;**  **default:**  **break;**  **}**  **}**  void main**()**  **{**  NETDEV\_Init**();**  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_SetExceptionCallBack**(**ExceptionCallBack\_PF**,NULL);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**    NETDEV\_Cleanup**();**  **return;**  **}**    NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  Sleep**(**120000**);**  NETDEV\_Logout**(**lUserID**);**  NETDEV\_Cleanup**();**  **}** |

## Process of Alarm Module

### Flowchart



### Sample code

|  |
| --- |
| The following C++ sample code shows the alarm process. For reference only. |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void STDCALL AlarmMessCallBack\_PF**(**IN LPVOID lpUserID**,**  IN LPNETDEV\_REPORT\_INFO\_S pstReportInfo**,**  IN LPVOID lpBuf**,**  IN INT32 dwBufLen**,**  IN LPVOID lpUserData  **)**  **{**  **switch(**pstReportInfo**->**stAlarmInfo**.**dwAlarmType**)**  **{**  **case** NETDEV\_ALARM\_MOVE\_DETECT**:**  printf**(**"Motion detection alarm\n"**);**  **break;**  **case** NETDEV\_ALARM\_MOVE\_DETECT\_RECOVER**:**  printf**(**"Motion detection alarm recover\n"**);**  **break;**  **default:**  **break;**  **}**  **}**  void main**()**  **{**    NETDEV\_Init**();**    NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**    NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**    NETDEV\_Cleanup**();**  **return;**  **}**    NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  bRet **=** NETDEV\_SetAlarmCallBack\_V30**(**lUserID**,** AlarmMessCallBack\_PF**,** **NULL);**  **if** **(**FALSE **==** bRet**)**  **{**  printf**(**"Set alarm callback error\n"**);**  **}**  **else**  **{**  printf**(**"Set alarm callback Success\n"**);**  **}**  Sleep**(**120000**);**  NETDEV\_Logout**(**lUserID**);**  NETDEV\_Cleanup**();**  **}** |

## Process of PTZ Module

### PTZ control process

#### Flowchart



#### Sample code

|  |
| --- |
| The following C++ sample code shows the PTZ control process. For reference only. |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void main**()**  **{**    NETDEV\_Init**();**    NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  NETDEV\_Cleanup**();**  **return;**  **}**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  INT32 dwChannelID **=** 1**;**    HWND hWnd **=** GetConsoleWindow**();**  NETDEV\_PREVIEWINFO\_S stNetInfo **=** **{**0**};**  stNetInfo**.**dwChannelID **=** dwChannelID**;**  stNetInfo**.**hPlayWnd **=** hWnd**;**  stNetInfo**.**dwStreamType **=** 0**;**  stNetInfo**.**dwLinkMode **=** 1**;**  LPVOID lpPlayHandle **=** NETDEV\_RealPlay**(**lUserID**,** **&**stNetInfo**,** **NULL,** 0**);**  **if** **(NULL** **==** lpPlayHandle**)**  **{**  printf**(**"RealPlay failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  **}**  NETDEV\_PTZControl\_Other**(**lUserID**,** dwChannelID**,** NETDEV\_PTZ\_TILTUP**,** 6**);**  Sleep**(**5000**);**  NETDEV\_PTZControl\_Other**(**lUserID**,** dwChannelID**,** NETDEV\_PTZ\_TILTDOWN**,** 6**);**  Sleep**(**5000**);**  NETDEV\_PTZControl\_Other**(**lUserID**,** dwChannelID**,** NETDEV\_PTZ\_PANRIGHT**,** 6**);**  Sleep**(**5000**);**  NETDEV\_PTZControl\_Other**(**lUserID**,** dwChannelID**,** NETDEV\_PTZ\_PANLEFT**,** 6**);**  Sleep**(**5000**);**  NETDEV\_PTZControl\_Other**(**lUserID**,** dwChannelID**,** NETDEV\_PTZ\_ALLSTOP**,** 6**);**  **if** **(NULL** **!=** lpPlayHandle**)**  **{**    NETDEV\_StopRealPlay**(**lpPlayHandle**);**  **}**    NETDEV\_Logout**(**lUserID**);**    NETDEV\_Cleanup**();**  **}** |

### Process of preset and patrol module

#### Flowchart



## Process of Smart Module

### Alarm process of person module

#### Flowchart



Flowchart of reporting person alarms



Flowchart of operating persons, person libraries and monitoring



Flowchart of searching face recognition records

#### Sample code for NVR or IPC reporting face recognition and comparison data

|  |
| --- |
| The following C++ sample code shows the face comparison process of NVR or IPC. For reference only. |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void STDCALL PersonAlarmCallBack**(**IN LPVOID lpUserID**,**IN LPNETDEV\_PERSON\_EVENT\_INFO\_S pstAlarmData**,**IN LPVOID lpUserData**)**  **{**  printf**(**"Person alarm callback\n"**);**  FILE**\*** fp**;**  **if** **(NULL** **!=** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPersonInfo**.**stImageInfo**[**0**].**stFileInfo**.**pcData **&&**  0 **!=** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPersonInfo**.**stImageInfo**[**0**].**stFileInfo**.**udwSize**)**  **{**  fp **=** fopen**(**"PersonImage.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPersonInfo**.**stImageInfo**[**0**].**stFileInfo**.**pcData**,** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPersonInfo**.**stImageInfo**[**0**].**stFileInfo**.**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **}**  **if** **(NULL** **!=** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPanoImage**.**pcData **&&**  0 **!=** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPanoImage**.**udwSize**)**  **{**  fp **=** fopen**(**"PanoImage.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPanoImage**.**pcData**,** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPanoImage**.**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **}**  **if** **(NULL** **!=** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stFaceImage**.**pcData **&&**  0 **!=** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stFaceImage**.**udwSize**)**  **{**  fp **=** fopen**(**"FaceImage.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stFaceImage**.**pcData**,** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stFaceImage**.**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **}**  **}**  void main**()**  **{**  NETDEV\_Init**();**  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**    NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**    NETDEV\_Cleanup**();**  **return;**  **}**    NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**    NETDEV\_SetPersonAlarmCallBack**(**lUserID**,**PersonAlarmCallBack**,**lUserID**);**  NETDEV\_LAPI\_SUB\_INFO\_S stSubInfo **=** **{**0**};**  NETDEV\_SUBSCRIBE\_SUCC\_INFO\_S stSubSuccInfo **=** **{**0**};**  stSubInfo**.**udwType **=** 0**;**  stSubInfo**.**udwLibIDNum **=** 0xffff**;**  bRet **=** NETDEV\_SubscibeLapiAlarm**(**lUserID**,** **&**stSubInfo**,** **&**stSubSuccInfo**);**  printf**(**"Subscibe lapi alarm: %d\n"**,** bRet**);**  Sleep**(**120000**);**  bRet **=** NETDEV\_UnSubLapiAlarm**(**lUserID**,** stSubSuccInfo**.**udwID**);**    NETDEV\_Logout**(**lUserID**);**    NETDEV\_Cleanup**();**  **}** |

#### Sample code for VMS reporting face comparison data

|  |
| --- |
| The following C++ sample code shows the face comparison process of VMS. For reference only. |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void STDCALL AlarmMessCallBack\_PF**(**IN LPVOID lpUserID**,**  IN LPNETDEV\_REPORT\_INFO\_S pstReportInfo**,**  IN LPVOID lpBuf**,**  IN INT32 dwBufLen**,**  IN LPVOID lpUserData  **)**  **{**  **switch(**pstReportInfo**->**stAlarmInfo**.**dwAlarmType**)**  **{**  **case** NETDEV\_ALARM\_MOVE\_DETECT**:**  printf**(**"Motion detection alarm\n"**);**  **break;**  **case** NETDEV\_ALARM\_MOVE\_DETECT\_RECOVER**:**  printf**(**"Motion detection alarm recover\n"**);**  **break;**  **default:**  **break;**  **}**  **}**  void STDCALL PersonAlarmCallBack**(**IN LPVOID lpUserID**,**IN LPNETDEV\_PERSON\_EVENT\_INFO\_S pstAlarmData**,**IN LPVOID lpUserData**)**  **{**  printf**(**"Person alarm callback\n"**);**  FILE**\*** fp**;**  **if** **(NULL** **!=** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPersonInfo**.**stImageInfo**[**0**].**stFileInfo**.**pcData **&&**  0 **!=** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPersonInfo**.**stImageInfo**[**0**].**stFileInfo**.**udwSize**)**  **{**  fp **=** fopen**(**"PersonImage.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPersonInfo**.**stImageInfo**[**0**].**stFileInfo**.**pcData**,** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPersonInfo**.**stImageInfo**[**0**].**stFileInfo**.**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **}**  **if** **(NULL** **!=** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPanoImage**.**pcData **&&**  0 **!=** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPanoImage**.**udwSize**)**  **{**  fp **=** fopen**(**"PanoImage.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPanoImage**.**pcData**,** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stPanoImage**.**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **}**  **if** **(NULL** **!=** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stFaceImage**.**pcData **&&**  0 **!=** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stFaceImage**.**udwSize**)**  **{**  fp **=** fopen**(**"FaceImage.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stFaceImage**.**pcData**,** pstAlarmData**->**stCtrlFaceInfo**->**stCompareInfo**.**stFaceImage**.**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **}**  **}**  void main**()**  **{**    NETDEV\_Init**();**    NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.108"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "123456"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_PRIVATE**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  NETDEV\_Cleanup**();**  **return;**  **}**  NETDEV\_SetAlarmCallBack\_V30**(**lUserID**,** AlarmMessCallBack\_PF**,** lUserID**);**  NETDEV\_SetPersonAlarmCallBack**(**lUserID**,**PersonAlarmCallBack**,**lUserID**);**  Sleep**(**120000**);**  NETDEV\_SetAlarmCallBack\_V30**(**lUserID**,** **NULL,** lUserID**);**  NETDEV\_Logout**(**lUserID**);**  NETDEV\_Cleanup**();**  **}** |

### Function process of people counting module

#### Flowchart



### Function process of vehicle module

#### Flowchart



Vehicle alarm flowchart



Vehicle, vehicle library, and monitoring flowchart



Flowchart of searching vehicle recognition records

#### Sample code for NVR or IPC reporting vehicle comparison data

|  |
| --- |
| The following C++ sample code shows the vehicle comparison process of NVR or IPC. For reference only. |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void STDCALL VehicleAlarmMessCallBack**(**IN LPVOID lpUserID**,**  IN LPNETDEV\_VEH\_RECOGNITION\_EVENT\_S pstVehicleAlarmInfo**,**  IN LPVOID lpBuf**,**  IN INT32 dwBufLen**,**  IN LPVOID lpUserData**)**  **{**  printf**(**"Vehicle alarm callback\n"**);**  FILE**\*** fp**;**  **for** **(**int i**=**0**;** i**<**pstVehicleAlarmInfo**->**stVehicleEventInfo**.**udwVehicleInfoNum**;** i**++)**  **{**  **if** **(NULL** **!=** pstVehicleAlarmInfo**->**stVehicleEventInfo**.**pstVehicleRecordInfo**[**i**].**stPlateImage**.**pcData**)**  **{**  fp **=** fopen**(**"Vehicle1.jpg"**,** "wb+"**);**  fwrite**(**pstVehicleAlarmInfo**->**stVehicleEventInfo**.**pstVehicleRecordInfo**[**i**].**stPlateImage**.**pcData**,** pstVehicleAlarmInfo**->**stVehicleEventInfo**.**pstVehicleRecordInfo**[**i**].**stPlateImage**.**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **}**  **if** **(NULL** **!=** pstVehicleAlarmInfo**->**stVehicleEventInfo**.**pstVehicleRecordInfo**[**i**].**stPanoImage**.**pcData**)**  **{**  fp **=** fopen**(**"Vehicle2.jpg"**,** "wb+"**);**  fwrite**(**pstVehicleAlarmInfo**->**stVehicleEventInfo**.**pstVehicleRecordInfo**[**i**].**stPanoImage**.**pcData**,** pstVehicleAlarmInfo**->**stVehicleEventInfo**.**pstVehicleRecordInfo**[**i**].**stPanoImage**.**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **}**  **if** **(NULL** **!=** pstVehicleAlarmInfo**->**stVehicleEventInfo**.**pstVehicleRecordInfo**[**i**].**stVehicleImage**.**pcData**)**  **{**  fp **=** fopen**(**"Vehicle3.jpg"**,** "wb+"**);**  fwrite**(**pstVehicleAlarmInfo**->**stVehicleEventInfo**.**pstVehicleRecordInfo**[**i**].**stVehicleImage**.**pcData**,** pstVehicleAlarmInfo**->**stVehicleEventInfo**.**pstVehicleRecordInfo**[**i**].**stVehicleImage**.**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **}**  **}**  **}**  void main**()**  **{**    NETDEV\_Init**();**    NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  NETDEV\_Cleanup**();**  **return;**  **}**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  NETDEV\_SetVehicleAlarmCallBack**(**lUserID**,**VehicleAlarmMessCallBack**,**lUserID**);**  NETDEV\_LAPI\_SUB\_INFO\_S stSubInfo **=** **{**0**};**  NETDEV\_SUBSCRIBE\_SUCC\_INFO\_S stSubSuccInfo **=** **{**0**};**  stSubInfo**.**udwType **=** 0**;**  stSubInfo**.**udwLibIDNum **=** 0xffff**;**  bRet **=** NETDEV\_SubscibeLapiAlarm**(**lUserID**,** **&**stSubInfo**,** **&**stSubSuccInfo**);**  printf**(**"Subscibe lapi alarm: %d\n"**,** bRet**);**  Sleep**(**120000**);**  bRet **=** NETDEV\_UnSubLapiAlarm**(**lUserID**,** stSubSuccInfo**.**udwID**);**  NETDEV\_Logout**(**lUserID**);**  NETDEV\_Cleanup**();**  **}** |

### Process of structured data (passing persons and vehicles)

#### Flowchart



#### Sample code for NVR or IPC reporting structured data

|  |
| --- |
| The following C++ sample code shows the structured data process of NVR or IPC. For reference only. |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void STDCALL StructAlarmCallBack**(**IN LPVOID lpUserID**,** IN LPNETDEV\_STRUCT\_ALARM\_INFO\_S pstAlarmInfo**,** IN LPNETDEV\_STRUCT\_DATA\_INFO\_S pstAlarmData**,** IN LPVOID lpUserData**)**  **{**  printf**(**"Struct alarm callback\n"**);**  FILE**\*** fp**;**  **if(NULL** **!=** pstAlarmData**->**stObjectInfo**.**pstFaceInfo **&&** 0 **<**pstAlarmData**->**stObjectInfo**.**udwFaceNum **&&** **NULL** **!=** pstAlarmData**->**pstImageInfo **&&** 0 **<** pstAlarmData**->**udwImageNum**)**  **{**  **for(**UINT32 i **=** 0**;** i **<** pstAlarmData**->**stObjectInfo**.**udwFaceNum**;** i**++)**  **{**  **for(**UINT32 j **=** 0**;**j **<** pstAlarmData**->**udwImageNum**;** j**++)**  **{**  **if(**pstAlarmData**->**stObjectInfo**.**pstFaceInfo**[**i**].**udwLargePicAttachIndex **==** pstAlarmData**->**pstImageInfo**[**j**].**udwIndex**)**  **{**  fp **=** fopen**(**"1.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**pstImageInfo**[**j**].**pszData**,** pstAlarmData**->**pstImageInfo**[**j**].**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **continue;**  **}**  **else** **if(**pstAlarmData**->**stObjectInfo**.**pstFaceInfo**[**i**].**udwSmallPicAttachIndex **==** pstAlarmData**->**pstImageInfo**[**j**].**udwIndex**)**  **{**  fp **=** fopen**(**"2.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**pstImageInfo**[**j**].**pszData**,** pstAlarmData**->**pstImageInfo**[**j**].**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **continue;**  **}**  **}**  **}**  **}**  **if(NULL** **!=** pstAlarmData**->**stObjectInfo**.**udwPersonNum **&&** 0 **<**pstAlarmData**->**stObjectInfo**.**udwPersonNum **&&** **NULL** **!=** pstAlarmData**->**pstImageInfo **&&** 0 **<** pstAlarmData**->**udwImageNum**)**  **{**  **for(**UINT32 i **=** 0**;** i **<** pstAlarmData**->**stObjectInfo**.**udwPersonNum**;** i**++)**  **{**  **for(**UINT32 j **=** 0**;**j **<** pstAlarmData**->**udwImageNum**;** j**++)**  **{**  **if(**pstAlarmData**->**stObjectInfo**.**pstPersonInfo**[**i**].**udwLargePicAttachIndex **==** pstAlarmData**->**pstImageInfo**[**j**].**udwIndex**)**  **{**  fp **=** fopen**(**"3.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**pstImageInfo**[**j**].**pszData**,** pstAlarmData**->**pstImageInfo**[**j**].**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **continue;**  **}**  **else** **if(**pstAlarmData**->**stObjectInfo**.**pstPersonInfo**[**i**].**udwSmallPicAttachIndex **==** pstAlarmData**->**pstImageInfo**[**j**].**udwIndex**)**  **{**  fp **=** fopen**(**"3.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**pstImageInfo**[**j**].**pszData**,** pstAlarmData**->**pstImageInfo**[**j**].**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **continue;**  **}**  **}**  **}**  **}**  **if(NULL** **!=** pstAlarmData**->**stObjectInfo**.**pstNonMotorVehInfo **&&** 0 **<**pstAlarmData**->**stObjectInfo**.**udwNonMotorVehNum **&&** **NULL** **!=** pstAlarmData**->**pstImageInfo **&&** 0 **<** pstAlarmData**->**udwImageNum**)**  **{**  **for(**UINT32 i **=** 0**;** i **<** pstAlarmData**->**stObjectInfo**.**udwNonMotorVehNum**;** i**++)**  **{**  **for(**UINT32 j **=** 0**;**j **<** pstAlarmData**->**udwImageNum**;** j**++)**  **{**  **if(**pstAlarmData**->**stObjectInfo**.**pstNonMotorVehInfo**[**i**].**udwLargePicAttachIndex **==** pstAlarmData**->**pstImageInfo**[**j**].**udwIndex**)**  **{**  fp **=** fopen**(**"4.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**pstImageInfo**[**j**].**pszData**,** pstAlarmData**->**pstImageInfo**[**j**].**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **continue;**  **}**  **else** **if(**pstAlarmData**->**stObjectInfo**.**pstNonMotorVehInfo**[**i**].**udwSmallPicAttachIndex **==** pstAlarmData**->**pstImageInfo**[**j**].**udwIndex**)**  **{**  fp **=** fopen**(**"5.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**pstImageInfo**[**j**].**pszData**,** pstAlarmData**->**pstImageInfo**[**j**].**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **continue;**  **}**  **}**  **}**  **}**  **if(NULL** **!=** pstAlarmData**->**stObjectInfo**.**pstVehInfo **&&** 0 **<**pstAlarmData**->**stObjectInfo**.**udwVehicleNum **&&** **NULL** **!=** pstAlarmData**->**pstImageInfo **&&** 0 **<** pstAlarmData**->**udwImageNum**)**  **{**  **for(**UINT32 i **=** 0**;** i **<** pstAlarmData**->**stObjectInfo**.**udwVehicleNum**;** i**++)**  **{**  **for(**UINT32 j **=** 0**;**j **<** pstAlarmData**->**udwImageNum**;** j**++)**  **{**  **if(**pstAlarmData**->**stObjectInfo**.**pstVehInfo**[**i**].**udwLargePicAttachIndex **==** pstAlarmData**->**pstImageInfo**[**j**].**udwIndex**)**  **{**  fp **=** fopen**(**"6.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**pstImageInfo**[**j**].**pszData**,** pstAlarmData**->**pstImageInfo**[**j**].**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **continue;**  **}**  **else** **if(**pstAlarmData**->**stObjectInfo**.**pstVehInfo**[**i**].**udwSmallPicAttachIndex **==** pstAlarmData**->**pstImageInfo**[**j**].**udwIndex**)**  **{**  fp **=** fopen**(**"6.jpg"**,** "wb+"**);**  fwrite**(**pstAlarmData**->**pstImageInfo**[**j**].**pszData**,**pstAlarmData**->**pstImageInfo**[**j**].**udwSize**,** 1**,** fp**);**  fclose**(**fp**);**  **continue;**  **}**  **}**  **}**  **}**  **return** **;**  **}**  void main**()**  **{**    NETDEV\_Init**();**    NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**    NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));**  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));**  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));**  stDevLoginInfo**.**dwPort **=** 80**;**  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;**  NETDEV\_SELOG\_INFO\_S stSELogInfo **=** **{**0**};**  LPVOID lUserID **=** NETDEV\_Login\_V30**(&**stDevLoginInfo**,** **&**stSELogInfo**);**  **if** **(NULL** **==** lUserID**)**  **{**  printf**(**"Login failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  NETDEV\_Cleanup**();**  **return;**  **}**    NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**  INT32 dwCount **=** 128**;**  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**\*** pstVideoChlDetailInfoEx **=** **new** NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**[**dwCount**];**  memset**(**pstVideoChlDetailInfoEx**,** 0**,** **sizeof(**NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S**)\***dwCount**);**  bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** pstVideoChlDetailInfoEx**);**  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**    NETDEV\_SetStructAlarmCallBack**(**lUserID**,**StructAlarmCallBack**,**lUserID**);**  NETDEV\_LAPI\_SUB\_INFO\_S stSubInfo **=** **{**0**};**  NETDEV\_SUBSCRIBE\_SUCC\_INFO\_S stSubSuccInfo **=** **{**0**};**  stSubInfo**.**udwType **=** 0**;**  stSubInfo**.**udwLibIDNum **=** 0xffff**;**  bRet **=** NETDEV\_SubscibeLapiAlarm**(**lUserID**,** **&**stSubInfo**,** **&**stSubSuccInfo**);**  printf**(**"Subscibe lapi alarm: %d\n"**,** bRet**);**  Sleep**(**120000**);**  bRet **=** NETDEV\_UnSubLapiAlarm**(**lUserID**,** stSubSuccInfo**.**udwID**);**  NETDEV\_Logout**(**lUserID**);**  NETDEV\_Cleanup**();**  **}** |

### Process of Smart Intrusion Prevention (SIP)

#### Event reporting flowchart



#### Object search flowchart



#### Note

* The perimeter protection type is differentiated by the rule type (udwRuleType) in the rule information (NETDEV\_RULE\_INFO\_S) for face, pedestrians, non-motor vehicles, and motor vehicles and object information (NETDEV\_OBJECT\_INFO\_S).
* SIP alarm process is applicable to IPC, NVR, and VMS.
* NVR and VMS support the object search process.
* Only VMS supports getting related data of specified alarms.
* If ordinary alarm events are not needed, there’s no need to call NETDEV\_SetAlarmCallBack\_V40.

### Function process of fire detection and thermal imaging

#### Flowchart



#### Note

* Thermal imaging alarm functions includes fire detection alarm, smoke alarm, and temperature alarm.
* Subscribe to fire detection alarms without images using Bit12, and register a callback function with NETDEV\_SetConflagrationAlarmCallBack to receive alarm data.
* Subscribe to fire detection alarms with images using Bit13, and register a callback function with NETDEV\_SetConflagrationAlarmCallBackV30 to receive alarm data with images. Subscribe to smoking detection alarms with images using NETDEV\_SetSmokeAlarmCallBack; subscribe to temperature detection alarms with images using NETDEV\_SetTemperatureDetcCallBack.
* Thermal imaging alarm process is applicable to IPC, AIBox, and VMS.

### High-rise littering and elevator entrance detection

#### Flowchart



#### Note

* Alarm recording search: NETDEV\_FindFile, NETDEV\_FindNextFile, NETDEV\_FindClose

Note: Among the recording search conditions for high-rise littering and elevator entrance alarms, thed wStreamType (recording storage type) should be 28 (high-rise littering) and 27 (elevator entrance detection), and dwRecordLocation (recording storage location) should be 1 (VMS storage).

* Interfaces for searching data related to specified alarms: NETDEV\_FindAlarmRelatedDataList, NETDEV\_FindNextAlarmRelatedDataInfo, NETDEV\_FindCloseAlarmRelatedDataList

Note: The alarm ID in the image search conditions is the alarm ID in the reported alarms, and the image information obtained is also in the structured data related to the alarms.

* This flowchart shows the process of integrating with VMS devices.

### People Counting

#### Flowchart



#### Note

* For details on login and getting channel info, please refer to 3.2 Major Process.
* If ordinary alarm events are not needed, there’s no need to call NETDEV\_SetAlarmCallBack\_V40.
* The current SDK has been used on IPC/VMS only, and the interface calling process is the same.
* On the IPC configuration page, people flow counting uses LineRuleData, and crowd density monitoring uses AreaRuleData.

# Basic Interface Definition

## SDK Initialization

### SDK initialization

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_Init(); |

**Interface description:**

Initialize SDK, which must be performed before calling other SDK functions.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* [NETDEV\_Init](#_SDK_初始化) needs to be called once within the program, and [NETDEV\_Cleanup](#_SDK清理) needs to be called once when logging out.

**See also:**

[NETDEV\_Cleanup](#_SDK清理)

### SDK cleanup

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ Cleanup (); |

**Interface description:**

SDK cleanup

**Return value:**

TRUE means success, other values mean failure. Call [[NETDEV\_GetLastError](#_获取接口错误码_1)](#_获取接口错误码) to get the error code and use the error code to determine the cause of error.

Remarks:

* No other SDK interfaces can be called when [NETDEV\_Cleanup](#_SDK清理) is being called.
* [NETDEV\_Init](#_SDK_初始化) and [NETDEV\_Cleanup](#_SDK清理) must be used in pairs.
* [NETDEV\_Init](#_SDK_初始化) needs to be called once within the program, and [NETDEV\_Cleanup](#_SDK清理) needs to be called once when logging out.

**See also:**

[NETDEV\_Init](#_SDK_初始化)

## SDK Local Functions

### Set log size and quantity

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ConfigLogFile  (  INT32 dwLogFileSize,  INT32 dwLogFileNum  ); |

**Interface description:**

Set the size and quantity of log files.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| dwLogFileSize | IN | Size of a log file. Unit: Byte. |
| dwLogFileNum | IN | Number of log files. |

**Return value:**

TRUE means success, other values mean failure.

Remarks:

* This interface must be called prior to SDK initialization ([NETDEV\_Init](#_SDK初始化)).
* The default log file number is 10 and the size of a log file is 30M.

**See also:**

[NETDEV\_SetLogPath](#_设置日志文件路径), [NETDEV\_SetWriteLogFlag](#_设置写入日志的标记)

### Set log file path

**Interface name:**

|  |
| --- |
| BOOL    STDCALL   NETDEV\_SetLogPath  (  CHAR \*       pszLogPath  ); |

**Interface description:**

Set the path of log files.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| pszLogPath | IN | Log path (not including filename). |

**Return value:**

TRUE means success, other values mean failure.

Remarks:

* This interface must be called prior to SDK initialization ([NETDEV\_Init](#_SDK初始化)).
* The default path is the path where the program runs.
* Make sure the log path you want to set already exists.

**See also:**

[NETDEV\_ConfigLogFile](#_设置日志文件大小和数量), [NETDEV\_SetWriteLogFlag](#_设置写入日志的标记)

### Set the flag for writing to logs

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetWriteLogFlag  (  BOOL bWriteLogFlag  ); |

**Interface description:**

Set the flag for writing to logs.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| bWriteLogFlag | IN | Flag for writing to logs. TRUE: Write to logs FALSE: Not write to logs |

**Return value:**

TRUE means success, other values mean failure.

Remarks:

* This interface must be called prior to SDK initialization ([NETDEV\_Init](#_SDK初始化)).
* Log writing is enabled by default and the flag is TRUE.

**See also:**

[NETDEV\_ConfigLogFile](#_设置日志文件大小和数量), [NETDEV\_SetLogPath](#_设置日志文件路径)

### Get SDK version

**Interface name:**

|  |
| --- |
| INT32 STDCALL NETDEV\_GetSDKVersion(); |

**Interface description:**

Get SDK version info.

**Return value:**

SDK version info.

Remarks:

* In the two high bytes, the high 8 bits means the major version, the low 8 bits means the minor version. The two low bytes means the additional version number. For example, 0x01080000 means version 1.8.0.0.

### Get interface error code

**Interface name:**

|  |
| --- |
| INT32        STDCALL NETDEV\_GetLastError(); |

**Interface description:**

Get the corresponding error code after calling an interface failed.

**Return value:**

The returned value is the error code. Error codes are mainly divided into common error, media related error, PTZ module error, playback module error, etc. See [Error Code List](#_错误码列表_1).

Remarks:

* This interface can be called to get the error code only when an interface returns failure.

### Register the callback function to receive exceptions

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetExceptionCallBack  (  NETDEV\_ExceptionCallBack\_PF cbExceptionCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive exception or reconnection messages.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| cbExceptionCallBack | IN | Callback function for receiving exception messages, which calls back exception info. |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_ExceptionCallBack\_PF)  (  LPVOID        lpUserID,  INT32           dwType,  LPVOID       lpExpHandle,  LPVOID        lpUserData,  [LPNETDEV\_EXCEPTION\_OUTPUT\_INFO\_S](#_播放器异常输出信息) pstExceptionOutputInfo  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwType | IN | Exception or reconnect message type. See  [NETDEV\_EXCEPTION\_TYPE\_E](#_异常回调的消息类型枚举) |
| lpExpHandle | IN | Corresponding exception handle. |
| lpUserData | IN | User data. Corresponds to the lpUserData passed in when registering callback. |
| pstExceptionOutputInfo | IN | Exception output info. Used by player only. See [NETDEV\_EXCEPTION\_OUTPUT\_INFO\_S](#_播放器异常输出信息). |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Set the maximum number of decoding channels

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetMaxDecChlNum  (  UINT32 dwMaxdecNum  ) ; |

**Interface description:**

Set the maximum number of decoding channels.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| dwMaxdecNum | IN | Maximum number of decoding channels. |

**Return value:**

TRUE means success, other values mean failure.

Remarks:

* This interface must be called prior to SDK initialization ([NETDEV\_Init](#_SDK初始化)).
* A failed call means 128 decoding channels.
* SDK supports up to 128 decoding channels.
* The number of decoding channels is the total number of decoding channels for live view, playback, and recording download.

### Set keep-alive interval and times

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetConnectTime  (  INT32 dwWaitTime,  INT32 dwTrytimes  ); |

**Interface description:**

Set keep-alive parameters.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| dwWaitTime | IN | Wait time interval (unit: second). |
| dwTrytimes | IN | Number of reconnecting attempts |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Both dwWaitTime and dwTrytimes must be greater than 0.

### Set receiving timeout

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetRevTimeOut  (  [LPNETDEV\_REV\_TIMEOUT\_S](#_超时时间结构体) pstRevTimeout  ); |

**Interface description:**

Set a timeout for SDK interfaces.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| pstRevTimeout | IN | Timeout. See [NETDEV\_REV\_TIMEOUT\_S](#_超时时间结构体). |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* The timeout for network messages must be greater than or equal to 1 second.
* The timeout for file transfer must be greater than or equal to 30 seconds.

### Set local listening address

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetListenAddr  (  INT32 dwListenType,  CHAR\* pszAddr  ); |

**Interface description:**

Set local listening address

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| dwListenType | IN | Listen function type. See [NETDEV\_LISTEN\_TYPE\_E](#_监听类型枚举) |
| pszAddr | IN | Listening address |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* This interface must be called prior to SDK initialization ([NETDEV\_Init](#_SDK初始化)).

## Device Discovery

### Register the callback function to discover devices

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetDiscoveryCallBack  (  NETDEV\_DISCOVERY\_CALLBACK\_PF cbDiscoveryCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to discover devices

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| cbDiscoveryCallBack | IN | Callback function for device discovery |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_DISCOVERY\_CALLBACK\_PF)  (  [LPNETDEV\_DISCOVERY\_DEVINFO\_S](#_设备发现的设备信息结构体) pstDevInfo,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| pstDevInfo | IN | Device info. See [NETDEV\_DISCOVERY\_DEVINFO\_S](#_设备发现的设备信息结构体) |
| lpUserData | IN | User data. Corresponds to the lpUserData passed in when registering callback. |

**Return value:**

0 means failure, other values mean user ID. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* This interface must be called prior to SDK initialization ([NETDEV\_Init](#_设备搜索)

**See also:**

[NETDEV\_Discovery](#_设备搜索)

### Discover devices

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_Discovery  (  CHAR \*pszBeginIP,  CHAR \*pszEndIP  ); |

**Interface description:**

Discover devices

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| pszBeginIP | IN | Start IP address |
| pszEndIP | IN | End IP address |

**Return value:**

0 means failure, other values mean user ID. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* It means automatic search when both pszBeginIP and pszEndIP are “0.0.0.0”; otherwise, it means search by network segment.
* First register the device discovery callback function, and then call this interface to discover devices. The discovered device info will be shown in callback.

**See also:**

[NETDEV\_SetDiscoveryCallBack](#_设置设备搜索回调函数)

## Configure No Login

### Modify device IP address

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyDeviceAddr([LPNETDEV\_DEV\_ADDR\_INFO\_S](#_设备地址信息结构体) pstDevAddrInfo); |

**Interface description:**

Modify device IP address (without login)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| pstDevAddrInfo | IN | Address of the device to be modified |

**Return value:**

0 means failure, other values mean user ID. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Modify device IP address without login.

**See also:**

[NETDEV\_SetDiscoveryCallBack](#_设置设备搜索回调函数), [NETDEV\_Discovery](#_设备搜索)

## User Login

### User logs into the device

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_Login\_V30  (  [LPNETDEV\_DEVICE\_LOGIN\_INFO\_S](#_设备登录信息结构体)   pstDevLoginInfo,  [LPNETDEV\_SELOG\_INFO\_S](#_安全登录信息结构体)             pstSELogInfo  ); |

**Interface description:**

User logs in to device

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| pstDevLoginInfo | IN | Device login info |
| pstSELogInfo | OUT | Secure login info |

**Return value:**

0 means failure, other values mean user ID. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* For local logins, SDK uses the same port as the one for accessing the device on webpage. The default port number is 80.
* pstSELogInfo is only applicable to devices that are logged in to using the private protocol, for example, VMS.
* NVR and IPC only support login via Onvif, and VMS only supports login via the private protocol.

**See also:**

[NETDEV\_Logout](#_用户注销)

### User logout

**Interface name:**

|  |
| --- |
| BOOL STDCALL     NETDEV\_Logout  (  LPVOID    lpUserID  ) ; |

**Interface description:**

User logout

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |

**Return value:**

0 means failure, other values mean user ID. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_Login\_V30](#_用户设备登录)

## Live View

### Live view

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_RealPlay(  LPVOID lpUserID,  [LPNETDEV\_PREVIEWINFO\_S](#_实况预览参数结构体) pstPreviewInfo,  NETDEV\_SOURCE\_DATA\_CALLBACK\_PF cbPlayDataCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Start live view

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstPreviewInfo | IN | Live view parameter. See [NETDEV\_PREVIEWINFO\_S](#_实况预览参数结构体) |
| cbPlayDataCallBack | IN | Stream data callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL \*NETDEV\_SOURCE\_DATA\_CALLBACK\_PF)  (  LPVOID lpPlayHandle,  const BYTE \*pucBuffer,  INT32 dwBufSize,  INT32 dwMediaDataType,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Current live view handle |
| pucBuffer | IN | Buffer storing unassembled media stream data |
| dwBufSize | IN | Buffer size |
| dwMediaDataType | IN | Media data type. See [NETDEV\_MEDIA\_DATA\_FORMAT\_E](#_媒体数据流格式) |
| lpUserParam | IN | User parameter, which is the lpUserData passed in when user calls the NETDEV\_RealPlay interface. |

**Return value:**

Stream startup handle. 0 means failure, other values are stream startup handles. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Cannot execute in the cbPlayDataCallBack callback function. It may take a long time.
* The callback function in this interface can be null, and so this function will not call back stream data for user. But user can still use the [NETDEV\_SetPlayParseCB](#_注册拼帧后视频数据回调) and [NETDEV\_SetPlayDecodeVideoCB](#_注册解码后视频数据回调) interfaces to register the callback function to receive stream data.

**See also:**

[NETDEV\_SetPlayParseCB](#_注册拼帧后视频数据回调), [NETDEV\_SetPlayDecodeVideoCB](#_注册解码后视频数据回调), [NETDEV\_StopRealPlay](#_停止实时预览)

### Stop live view

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopRealPlay  (  LPVOID lpPlayHandle  ); |

**Interface description:**

Stop live view

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle, return value from the [NETDEV\_RealPlay](#_实时预览) interface |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_RealPlay](#_实时预览), [NETDEV\_FastRealPlayByUrl](#_URL起流)

### Start local recording

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SaveRealData  (  LPVOID lpPlayHandle,  CHAR \*pszSaveFileName,  INT32 dwFormat  ); |

**Interface description:**

Start local recording

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |
| pszSaveFileName | IN | Saved file name |
| dwFormat | IN | File format. See [NETDEV\_MEDIA\_FILE\_FORMAT\_E](#_媒体文件格式枚举) |

Remarks:

* Stream startup handles can be handles for live view and playback.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_StopSaveRealData](#_停止本地录像)

### Stop local recording

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopSaveRealData  (  LPVOID lpPlayHandle  ); |

**Interface description:**

Stop local recording

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |

Remarks:

* Stream startup handles can be handles for live view and playback.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SaveRealData](#_本地录像)

### Get live view URL

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetStreamUrl  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwStreamType,  CHAR \*pszStreamUrl  ); |

**Interface description:**

Start live view

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwChannelID | IN | Channel ID |
| dwStreamType | IN | Stream type. See [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举). |
| pszStreamUrl | OUT | Stream URL. User allocates RAM. The size must be greater than or equal to 260. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* The pszStreamUrl parameter needs user to assign RAM. The size must be greater than or equal to 260.

**See also:**

[NETDEV\_FastRealPlayByUrl](#_URL起流_1)

### Start live view by URL

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FastRealPlayByUrl  (  LPVOID lpUserID,  CHAR \*pszUrl,  [LPNETDEV\_PREVIEWINFO\_](#_实况预览参数结构体)S pstPreviewInfo,  NETDEV\_SOURCE\_DATA\_CALLBACK\_PF cbPlayDataCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Start live view by URL

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pszUrl | IN | Stream URL, which is obtained using [NETDEV\_GetStreamUrl](#_获取起流URL) |
| pstPreviewInfo | IN | Live view parameter. See [NETDEV\_PREVIEWINFO\_S](#_实况预览参数结构体) |
| cbPlayDataCallBack | IN | Stream data callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL \*NETDEV\_SOURCE\_DATA\_CALLBACK\_PF)  (  LPVOID lpPlayHandle,  const BYTE \*pucBuffer,  INT32 dwBufSize,  INT32 dwMediaDataType,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Current live view handle |
| pucBuffer | IN | Buffer storing unassembled media stream data |
| dwBufSize | IN | Buffer size |
| dwMediaDataType | IN | Media data type. See [NETDEV\_MEDIA\_DATA\_FORMAT\_E](#_媒体数据流格式) |
| lpUserParam | IN | User parameter, which is the lpUserData passed in when user calls the NETDEV\_RealPlay interface. |

**Return value:**

Stream startup handle. 0 means failure, other values are stream startup handles. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Cannot execute in the cbPlayDataCallBack callback function. It may take a long time.
* The callback function in this interface can be null, and so this function will not call back stream data for user. But user can still use the [NETDEV\_SetPlayParseCB](#_注册拼帧后视频数据回调) and [NETDEV\_SetPlayDecodeVideoCB](#_注册解码后视频数据回调) interfaces to register the callback function to receive stream data.

**See also:**

[NETDEV\_SetPlayParseCB](#_注册拼帧后视频数据回调), [NETDEV\_SetPlayDecodeVideoCB](#_注册解码后视频数据回调), [NETDEV\_StopRealPlay](#_停止实时预览), [NETDEV\_GetStreamUrl](#_URL起流), [NETDEV\_FastRealPlayByUrl](#_URL起流)

### Snapshot in preview

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_CapturePicture  (  LPVOID lpPlayHandle,  char \*pszFileName,  INT32 dwCaptureMode  ); |

**Interface description:**

Start live view by URL

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |
| pszFileName | IN | Path to the saved image file, including filename |
| dwCaptureMode | IN | Image format. See [NETDEV\_PICTURE\_FORMAT\_E](#_抓图图片格式枚举) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Stream startup handles can be handles for live view and playback.
* The image file path must include the filename.
* The filename must not include the suffix of the snapshot format.
* The snapshot file path must be existing path.

**See also:**

[NETDEV\_CaptureNoPreview](#_非预览抓图), [NETDEV\_CaptureNoPreviewV2](#_非预览获取图片数据)

### Snapshot without preview

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_CaptureNoPreview  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwStreamType,  CHAR \*pszFileName,  INT32 dwCaptureMode  ); |

**Interface description:**

Capture image without preview

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwChannelID | IN | Channel ID |
| dwStreamType | IN | Stream type. See [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举) |
| pszFileName | IN | Image file path, including the filename |
| dwCaptureMode | IN | Format of saved image. See [NETDEV\_PICTURE\_FORMAT\_E](#_抓图图片格式枚举). |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* The image file path must include the filename.
* The filename must not include the suffix of the snapshot format.
* Currently only JPG format is supported.
* The snapshot file path must already exist.

**See also:**

[NETDEV\_CapturePreview](#_预览抓图), [NETDEV\_CaptureNoPreviewV2](#_非预览获取图片数据)

### Get image data without preview

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_CaptureNoPreviewV2  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwStreamType,  INT32 dwCaptureMode,  INT32 dwPicBufferSize,  CHAR \*pszPicBuffer,  INT32 \*pdwPicRealSize  ); |

**Interface description:**

Start live view by URL

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwChannelID | IN | Channel ID |
| dwStreamType | IN | Stream type. See [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举). |
| dwCaptureMode | IN | Format of saved image. See [NETDEV\_PICTURE\_FORMAT\_E](#_抓图图片格式枚举) |
| dwPicBufferSize | IN | Size of buffer for receiving image data. |
| pszPicBuffer | OUT | Buffer for receiving image data. |
| pdwPicRealSize | OUT | Actual size of image data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Buffer for receiving image data. User allocates RAM size and assigns the buffer size to dwPicBufferSize.
* Currently only JPG format is supported.

**See also:**

[NETDEV\_CapturePreview](#_预览抓图), [NETDEV\_CaptureNoPreview](#_非预览抓图)

### Set recording file size

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetRecordSize  (  LPVOID lpPlayHandle,  UINT32 ulFileSize  ); |

**Interface description:**

Set recording file size

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Play handle |
| ulFileSize | IN | Recording file size (MB). Range: 10MB-1024MB |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

## Remote Playback

### Search recording

#### Search recording file list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindFile  (  LPVOID lpUserID,  [LPNETDEV\_FILECOND\_S](#_录像查找结构体) pstFindCond  ); |

**Interface description:**

Search recording files by file type and time

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstFindCond | IN | Recording search criteria |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextFile](#_逐个获取查找到的文件信息) and [NETDEV\_FindClose](#_关闭录像文件查找).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* This interface is used with the [NETDEV\_FindFile](#_逐个获取查找到的文件信息) and [NETDEV\_FindClose](#_关闭录像文件查找) interfaces.
* After calling this interface successfully, call the [NETDEV\_FindNextFile](#_逐个获取查找到的文件信息) interface repeatedly to get the next info.
* The [NETDEV\_FindClose](#_关闭录像文件查找) interface must be called to release resource and stop searching after recording file info is obtained.

**See also:**

[NETDEV\_FindNextFile](#_逐个获取查找到的文件信息), [NETDEV\_FindClose](#_关闭录像文件查找)

#### Get file info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextFile  (  LPVOID lpFindHandle,  [LPNETDEV\_FINDDATA\_S](#_录像文件信息结构体) pstFindData  ); |

**Interface description:**

Get recording file info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | File search handle, which is the return value from the [NETDEV\_FindFile](#_查找录像文件列表) interface |
| pstFindData | OUT | Recording file info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* This interface is used with the [NETDEV\_FindFile](#_查找录像文件列表) and [NETDEV\_FindClose](#_关闭录像文件查找) interfaces.
* Call this interface repeatedly to get the next info.
* The [NETDEV\_FindClose](#_关闭录像文件查找) interface must be called to release resource and stop searching after recording file info is obtained.

**See also:**

[NETDEV\_FindFile](#_查找录像文件列表), [NETDEV\_FindClose](#_关闭录像文件查找)

#### Stop searching recording file

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindClose  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching recording file, release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | File search handle, which is the return value from the [NETDEV\_FindFile](#_查找录像文件列表) interface |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* This interface is used with the [NETDEV\_FindFile](#_查找录像文件列表) and [NETDEV\_FindNextFile](#_逐个获取查找到的文件信息) interfaces.
* The [NETDEV\_FindClose](#_关闭录像文件查找) interface must be called to release resource and stop searching after recording file info is obtained.

**See also:**

[NETDEV\_FindFile](#_查找录像文件列表), [NETDEV\_FindNextFile](#_逐个获取查找到的文件信息)

### Get recording days

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetVideoDayNums  (  LPVOID lpUserID,  INT32 dwChannelID,  UINT32 \*dwDayNums  ); |

**Interface description:**

Search the number of recording days of a channel.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwChannelID | IN | Channel ID |
| dwDayNums | OUT | Number of recording days |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_QuickSearch](#_单通道查询), [NETDEV\_QuickSearchMultiChl](#_多通道查询)

### Search the recording distribution info of a specified month

#### Sing-channel search

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_QuickSearch  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_MONTH\_INFO\_S](#_按月查询录像分布信息结构体) pstMonthInfo,  [LPNETDEV\_MONTH\_STATUS\_S](#_录像分布状态结构体) pstMonthStatus  ); |

**Interface description:**

Search the recording distribution info in a specified month

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwChannelID | IN | Channel ID |
| pstMonthInfo | IN | Month info |
| pstMonthStatus | OUT | Recording distribution status |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* This interface can only search recording distribution info of one channel.

**See also:**

[NETDEV\_GetVideoDayNums](#_获取录像天数), [NETDEV\_QuickSearchMultiChl](#_多通道查询)

#### Multi-channel search

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_QuickSearchMultiChl  (  LPVOID lpUserID,  [LPNETDEV\_CHANNELS\_S](#_通道列表结构体) pstChannels,  [LPNETDEV\_MONTH\_INFO\_S](#_按月查询录像分布信息结构体) pstMonthInfo,  [LPNETDEV\_MONTH\_STATUS\_S](#_录像分布状态结构体) pstMonthStatus  ); |

**Interface description:**

Search the recording distribution info of multiple channels in a specified month

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstChannels | IN | Channel list |
| pstMonthInfo | IN | Month info |
| pstMonthStatus | OUT | Recording distribution status |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_QuickSearch](#_单通道查询), [NETDEV\_GetVideoDayNums](#_获取录像天数)

### Get playback URL

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetReplayUrl\_V30  (  LPVOID lpUserID,  [LPNETDEV\_PLAYBACKCOND\_S](#_录像回放参数结构体) pstPlayBackCond,  CHAR\* pszUrl  ); |

**Interface description:**

Get playback URL

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstPlayBackCond | IN | Recording playback parameter |
| pszUrl | OUT | Stream URL. RAM size must be greater than or equal to 260. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_FastPlayBackByUrl](#_根据URL回放录像文件)

### Play recording file by URL

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FastPlayBackByUrl  (  LPVOID lpUserID,  CHAR \*pszUrl,  [LPNETDEV\_PLAYBACKCOND\_S](#_录像回放参数结构体) pstPlayBackParam  ); |

**Interface description:**

Play recording file by URL

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pszUrl | IN | Stream URL |
| pstPlayBackParam | IN | Structure of recording playback parameters |

**Return value:**

Stream startup handle. 0 means failure, other values are stream startup handles. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetReplayUrl](#_获取回放URL)\_V30

### Play recording file by filename

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_PlayBackByName  (  LPVOID lpUserID,  [LPNETDEV\_PLAYBACKINFO\_S](#_按文件名回放录像参数结构体) pstPlayBackInfo  ); |

**Interface description:**

Play recording file by filename

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstPlayBackInfo | IN | Recording playback info |

**Return value:**

Stream startup handle. 0 means failure, other values are stream startup handles. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PlayBackByTime](#_按时间回放录像文件)

### Play recording file by time

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_PlayBackByTime  (  LPVOID lpUserID,  [LPNETDEV\_PLAYBACKCOND\_S](#_录像回放参数结构体) pstPlayBackInfo  ); |

**Interface description:**

Play recording file by time

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstPlayBackInfo | IN | Recording playback info |

**Return value:**

Stream startup handle. 0 means failure, other values are stream startup handles. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PlayBackByName](#_按文件名回放录像文件)

### Playback by frame

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetOneFramePlayWithTime  (  LPVOID lpPlayHandle,  INT64 dwPlayTime  ); |

**Interface description:**

Playback by frame at a specified time

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Playback handle |
| dwPlayTime | IN | Specified time |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PlayBackByName](#_按文件名回放录像文件), [NETDEV\_PlayBackByTime](#_按时间回放录像文件), [NETDEV\_FastPlayBackByUrl](#_根据URL回放录像文件)

### Playback control

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PlayBackControl  (  LPVOID lpPlayHandle,  INT32 dwControlCode,  LPVOID lpBuffer  ); |

**Interface description:**

Control playback status

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Playback handle |
| dwControlCode | IN | Playback control command. See [NETDEV\_VOD\_PLAY\_CTRL\_E](#_回放控制命令枚举) |
| lpBuffer | INOUT | Input/output parameter of playback control. If it is playback speed, see [NETDEV\_VOD\_PLAY\_STATUS\_E](#_播放/下载速度枚举) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**Remarks:**

* When starting, pausing or resuming playback, set IpBuffer to NULL.

**See also:**

[NETDEV\_PlayBackByName](#_按文件名回放录像文件), [NETDEV\_PlayBackByTime](#_按时间回放录像文件), [NETDEV\_FastPlayBackByUrl](#_根据URL回放录像文件)

### Stop playback

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopPlayBack  (  LPVOID lpPlayHandle  ); |

**Interface description:**

Stop playback

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Playback handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PlayBackByName](#_按文件名回放录像文件), [NETDEV\_PlayBackByTime](#_按时间回放录像文件), [NETDEV\_FastPlayBackByUrl](#_根据URL回放录像文件)

### Download recording file by filename

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_GetFileByName  (  LPVOID lpUserID,  [LPNETDEV\_PLAYBACKINFO\_S](#_按文件名回放录像参数结构体) pstPlayBackInfo,  CHAR \*pszSaveFileName,  INT32 dwFormat  ); |

**Interface description:**

Download recording file by filename

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstPlayBackInfo | IN | Recording playback parameter |
| pszSaveFileName | IN | File path (absolute path, including filename) |
| dwFormat | IN | Recording file format. See [NETDEV\_MEDIA\_FILE\_FORMAT\_E](#_媒体文件格式枚举) |

**Return value:**

Stream startup handle. 0 means failure, other values are stream startup handles. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* The file path must be an absolute path and include the filename.
* The snapshot file path must already exist.

**See also:**

[NETDEV\_GetFileByTime](#_按时间下载录像文件), [NETDEV\_StopGetFile](#_停止下载录像文件)

### Download recording file by time

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_GetFileByTime  (  LPVOID lpUserID,  [LPNETDEV\_PLAYBACKCOND\_S](#_录像回放参数结构体) pstPlayBackCond,  CHAR \*pszSaveFileName,  INT32 dwFormat  ); |

**Interface description:**

Download recording file by time

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstPlayBackCond | IN | Recording playback parameter |
| pszSaveFileName | IN | File path (absolute path, including filename) |
| dwFormat | IN | Recording file format. See [NETDEV\_MEDIA\_FILE\_FORMAT\_E](#_媒体文件格式枚举) |

**Return value:**

Stream startup handle. 0 means failure, other values are stream startup handles. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* The file path must be an absolute path and include the filename.
* The snapshot file path must already exist.

**See also:**

[NETDEV\_GetFileByName](#_按文件名下载录像文件), [NETDEV\_StopGetFile](#_停止下载录像文件)

### Stop downloading recording file

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopGetFile  (  LPVOID lpPlayHandle  ); |

**Interface description:**

Stop downloading recording file

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Recording download handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetFileByName](#_按文件名下载录像文件), [NETDEV\_GetFileByTime](#_按时间下载录像文件)

### Search recording start and end times

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_QueryRecordRange  (  LPVOID lpUserID,  [LPNETDEV\_CHANNEL\_LIST\_S](#_通道信息列表) pstChlList,  [LPNETDEV\_RECORD\_TIME\_LIST\_S](#_录像时间列表) pstRecordTimeList  ); |

**Interface description:**

Search recording start and end times

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstChlList | IN | Channel list |
| pstRecordTimeList | OUT | Recording time list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

## Stream Data Callback

### Register the callback function to receive raw streams

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPlayDataCallBack  (  LPVOID lpPlayHandle,  NETDEV\_SOURCE\_DATA\_CALLBACK\_PF cbPlayDataCallBack,  INT32 bContinue,  LPVOID lpUser  ); |

**Interface description:**

Register the callback function to receive raw streams

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| cbPlayDataCallBack | IN | Stream data callback function |
| bContinue | IN | Whether to continue the subsequent assembly, decode and display operations |
| lpUser | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL \*NETDEV\_SOURCE\_DATA\_CALLBACK\_PF)  (  LPVOID lpPlayHandle,  const BYTE \*pucBuffer,  INT32 dwBufSize,  INT32 dwMediaDataType,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Current live view handle/playback handle |
| pucBuffer | IN | Buffer storing unassembled media stream data |
| dwBufSize | IN | Buffer size |
| dwMediaDataType | IN | Media data type. See [NETDEV\_MEDIA\_DATA\_FORMAT\_E](#_媒体数据流格式) |
| lpUserParam | IN | User data. User parameter specified when calling the NETDEV\_SetPlayDataCallBack function. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* To close the callback function, set the second parameter to NULL.
* User shall process the output data in time to ensure that the function is returned soon enough; otherwise, media stream processing within the player will be affected.

**See also:**

[NETDEV\_RealPlay](#_实时预览), [NETDEV\_FastRealPlayByUrl](#_通过URL启动实况预览)

### Register the callback function to receive decoded audio data

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPlayDecodeAudioCB  (  LPVOID lpPlayHandle,  NETDEV\_DECODE\_AUDIO\_DATA\_CALLBACK\_PF cbPlayDecodeAudioCallBack,  INT32 bContinue,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive the decoded audio stream data

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| cbPlayDecodeAudioCallBack | IN | Stream data callback function |
| bContinue | IN | Whether to continue the play operation |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL\* NETDEV\_DECODE\_AUDIO\_DATA\_CALLBACK\_PF)  (  LPVOID lpPlayHandle,  const [NETDEV\_WAVE\_DATA\_S](#_音频数据结构体) \*pstWaveData,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Current live view handle/playback handle |
| pstWaveData | IN | Buffer storing info about decoded audio stream data. See [NETDEV\_WAVE\_DATA\_S](#_音频数据结构体). |
| lpUserParam | IN | User data. User parameter specified when calling NETDEV\_SetPlayDecodeAudioCB. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* To close the callback function, set the second parameter to NULL.
* User shall process the output data in time to ensure that the function is returned soon enough; otherwise, media stream processing within the player will be affected.

**See also:**

[NETDEV\_RealPlay](#_实时预览), [NETDEV\_FastRealPlayByUrl](#_通过URL启动实况预览)

### Register the callback function to receive decoded video data

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPlayDecodeVideoCB  (  LPVOID lpPlayHandle,  NETDEV\_DECODE\_VIDEO\_DATA\_CALLBACK\_PF cbPlayDecodeVideoCALLBACK,  INT32 bContinue,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive decoded video stream data.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| cbPlayDecodeVideoCALLBACK | IN | Stream data callback function |
| bContinue | IN | Whether to continue the subsequent display operation |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL\* NETDEV\_DECODE\_VIDEO\_DATA\_CALLBACK\_PF)  (  LPVOID lpPlayHandle,  const [NETDEV\_PICTURE\_DATA\_S](#_图像数据结构体) \*pstPictureData,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Current live view handle/playback handle |
| pstPictureData | IN | Buffer storing info about decoded video stream data. See [NETDEV\_PICTURE\_DATA\_S](#_图像数据结构体) |
| lpUserParam | IN | User data. User parameter specified when calling NETDEV\_SetPlayDecodeVideoCB. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* To close the callback function, set the second parameter to NULL.
* User shall process the output data in time to ensure that the function is returned soon enough; otherwise, media stream processing within the player will be affected.

**See also:**

[NETDEV\_RealPlay](#_实时预览), [NETDEV\_FastRealPlayByUrl](#_通过URL启动实况预览)

### Register callback function to receive assembled audio data

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPlayParseAudioCB  (  LPVOID lpPlayHandle,  NETDEV\_PARSE\_AUDIO\_DATA\_CALLBACK\_PF cbPlayParseAudioCallBack,  INT32 bContinue,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive assembled audio stream data

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| cbPlayParseAudioCallBack | IN | Stream data callback function |
| bContinue | IN | Whether to continue the play operation |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL\* NETDEV\_PARSE\_AUDIO\_DATA\_CALLBACK\_PF)  (  LPVOID lpPlayHandle,  const [NETDEV\_PARSE\_AUDIO\_DATA\_S](#_拼帧后音频数据结构体) \*pstParseAudioData,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Current live view handle/playback handle |
| pstParseAudioData | IN | Buffer storing info about assembled audio stream data. See [NETDEV\_PARSE\_AUDIO\_DATA\_S](#_拼帧后音频数据结构体) |
| lpUserParam | IN | User data. User parameter specified when calling NETDEV\_SetPlayParseAudioCB. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* To close the callback function, set the second parameter to NULL.
* User shall process the output data in time to ensure that the function is returned soon enough; otherwise, media stream processing within the player will be affected.

**See also:**

[NETDEV\_RealPlay](#_实时预览), [NETDEV\_FastRealPlayByUrl](#_通过URL启动实况预览)

### Register callback function to receive assembled video data

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPlayParseCB  (  LPVOID lpPlayHandle,  NETDEV\_PARSE\_VIDEO\_DATA\_CALLBACK\_PF cbParsePlayDataCallBack,  INT32 bContinue,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive assembled video stream data

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| cbParsePlayDataCallBack | IN | Stream data callback function |
| bContinue | IN | Whether to continue the subsequent decode and display operations |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL \*NETDEV\_PARSE\_VIDEO\_DATA\_CALLBACK\_PF)  (  LPVOID lpPlayHandle,  const [NETDEV\_PARSE\_VIDEO\_DATA\_S](#_视频解析数据结构体) \*pstParseVideoData,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Current live view handle/playback handle |
| pstParseVideoData | IN | Buffer storing info about the assembled video stream data. See [NETDEV\_PARSE\_VIDEO\_DATA\_S](#_视频解析数据结构体) |
| lpUserParam | IN | User data. User parameter specified when calling NETDEV\_SetPlayParseCB. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* To close the callback function, set the second parameter to NULL.
* User shall process the output data in time to ensure that the function is returned soon enough; otherwise, media stream processing within the player will be affected.

**See also:**

[NETDEV\_RealPlay](#_实时预览), [NETDEV\_FastRealPlayByUrl](#_通过URL启动实况预览)

### Register image callback

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPlayDisplayCB  (  LPVOID lpPlayHandle,  NETDEV\_DISPLAY\_CALLBACK\_PF cbPlayDisplayCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive the displayed data

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| cbPlayDisplayCallBack | IN | Stream data callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL\* NETDEV\_DISPLAY\_CALLBACK\_PF)  (  LPVOID lpHandle,  LPVOID hdc,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpHandle | IN | Live view handle/playback handle |
| hdc | IN | HDC, which user can use to do drawing. |
| lpUserParam | IN | User parameter specified when calling the NETDEV\_ SetPlayDisplayCB function. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* To close the callback function, set the second parameter to NULL.
* User shall process the output data in time to ensure that the function is returned soon enough; otherwise, media stream processing within the player will be affected.

**See also:**

[NETDEV\_RealPlay](#_实时预览), [NETDEV\_FastRealPlayByUrl](#_通过URL启动实况预览)

## Video Parameter Configuration

### Get packet loss rate of window

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetLostPacketRate  (  LPVOID lpPlayHandle,  INT32 \*pulRecvPktNum,  INT32 \*pulLostPktNum  ); |

**Interface description:**

Get packet loss rate of window

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| pulRecvPktNum | OUT | Number of packets received |
| pulLostPktNum | OUT | Number of packets lost |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* The packet loss rate (number of packets lost/number of packets received) needs to be calculated by user.

**See also:**

[NETDEV\_ResetLostPacketRate](#_重置窗口丢包率)

### Reset packet loss rate of window

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ResetLostPacketRate  (  LPVOID lpPlayHandle  ); |

**Interface description:**

Reset packet loss rate of window

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetLostPacketRate](#_获取窗口丢包率)

### Get bit rate of window

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetBitRate  (  LPVOID lpPlayHandle,  INT32 \*pdwBitRate  ); |

**Interface description:**

Get bit rate of window

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| pdwBitRate | OUT | Current bit rate of the window |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get frame rate of window

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetFrameRate  (  LPVOID lpPlayHandle,  INT32 \*pdwFrameRate  ); |

**Interface description:**

Get frame rate of window

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| pdwFrameRate | OUT | Current frame rate of the window |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get window resolution

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetResolution  (  LPVOID lpPlayHandle,  INT32 \*pdwWidth,  INT32 \*pdwHeight  ); |

**Interface description:**

Get video resolution of the window

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| pdwWidth | OUT | Obtained resolution - width |
| pdwHeight | OUT | Obtained resolution - height |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get image parameters

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetVideoEffect  (  LPVOID lpPlayHandle,  [LPNETDEV\_VIDEO\_EFFECT\_S](#_影像信息结构体) pstImageInfo  ); |

**Interface description:**

Get current image parameters

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| pstImageInfo | OUT | Image parameter. See [NETDEV\_VIDEO\_EFFECT\_S](#_影像信息结构体) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Get current image parameters only

### Adjust image parameters

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetVideoEffect  (  LPVOID lpPlayHandle,  [LPNETDEV\_VIDEO\_EFFECT\_S](#_影像信息结构体) pstImageInfo  ); |

**Interface description:**

Adjust current image parameters

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| pstImageInfo | IN | Image parameters. See [NETDEV\_VIDEO\_EFFECT\_S](#_影像信息结构体) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Adjust image parameters (current image only)

### Get encoding format of window

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetVideoEncodeFmt  (  LPVOID lpPlayHandle,  INT32 \*pdwVideoEncFmt  ); |

**Interface description:**

Get encoding format of current window

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| pdwVideoEncFmt | OUT | Video encoding format. See [NETDEV\_VIDEO\_CODE\_TYPE\_E](#_视频编码格式枚举) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Generate a keyframe dynamically

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_MakeKeyFrame  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwStreamType  ); |

**Interface description:**

Generate a keyframe dynamically, that is, forced I frame.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwChannelID | IN | Channel ID |
| dwStreamType | IN | Stream type. See [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举). |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Set digital zoom

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetDigitalZoom  (  LPVOID lpPlayHandle,  LPVOID hWnd,  [LPNETDEV\_RECT\_S](#_矩形区域结构体) pstRect  ); |

**Interface description:**

Set digital zoom

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| hWnd | IN | Window handle |
| pstRect | IN | Rectangle area of digital zoom |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* When pstRect is null, the whole image will be displayed (exit digital zoom)

### Enable/disable metadata processing

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetIVAEnable  (  LPVOID lpPlayHandle,  BOOL bEnableIVA  ); |

**Interface description:**

Enable/disable metadata processing

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle |
| bEnableIVA | IN | Whether to add metadata |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SetIVAShowParam](#_设置元数据处理参数)

### Set metadata processing parameter

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetIVAShowParam  (  INT32 dwShowParam  ); |

**Interface description:**

Set metadata display types

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| dwShowParam | IN | Metadata display type. See [NETDEV\_IVA\_SHOW\_RULE\_E](#_元数据显示类型) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SetIVAEnable](#_设置元数据处理开关)

### Set video fluency

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPictureFluency  (  LPVOID lpPlayHandle,  INT32 dwFluency  ); |

**Interface description:**

Set video fluency

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| dwFluency | IN | Fluency priority type. See [NETDEV\_PICTURE\_FLUENCY\_E](#_图像播放流畅性枚举). |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Set image display scale

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetRenderScale  (  LPVOID lpPlayHandle,  [NETDEV\_RENDER\_SCALE\_E](#_视频显示比例枚举) enRenderScalesss  ); |

**Interface description:**

Set image display scale

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle/playback handle |
| enRenderScale | IN | Image display scale |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

## Audio Function

### Start two-way audio

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_StartVoiceCom  (  LPVOID lpUserID,  INT32 dwChannelID,  NETDEV\_DECODE\_AUDIO\_DATA\_CALLBACK\_PF cbPlayDataCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Start two-way audio

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwChannelID | IN | Channel ID |
| cbPlayDataCallBack | IN | Callback function for receiving decoded audio stream data |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL\* NETDEV\_DECODE\_AUDIO\_DATA\_CALLBACK\_PF)  (  LPVOID lpPlayHandle,  const [NETDEV\_WAVE\_DATA\_S](#_音频数据结构体) \*pstWaveData,  LPVOID lpUserParam  )； |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Handle returned by the two-way audio interface |
| pstWaveData | IN | Info about parsed audio stream data. See [NETDEV\_WAVE\_DATA\_S](#_音频数据结构体) |
| lpUserParam | IN | User parameter, which is the lpUserData passed in when user calls the NETDEV\_StartVoiceCom interface. |

**Return value:**

0 means failure, other values are used as handle of functions such as [NETDEV\_StopVoiceCom](#_停止语音对讲). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Implementation of cbPlayDataCallBack should not take too long.
* The callback function and user parameters in this function can be null.
* For the device itself, the channel ID is 0. For channels under the device, the channel IDs are queried by the SDK interface.

**See also:**

[NETDEV\_StopVoiceCom](#_停止语音对讲)

### Stop two-way audio

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopVoiceCom (LPVOID lpPlayHandle); |

**Interface description:**

Stop two-way audio

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Two-way audio handle, which is the return value from the [NETDEV\_StartVoiceCom](#_开启语音对讲) interface |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_StartVoiceCom](#_开启语音对讲)

### Enable audio data forwarding

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_StartInputVoiceSrv  (  LPVOID lpUserID,  INT32 dwChannelID  ); |

**Interface description:**

Enable audio data forwarding

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwChannelID | IN | Channel ID |

**Return value:**

0 means failure, other values are used as handle of functions such as [NETDEV\_StopInputVoiceSrv](#_设置解析后语音数据). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* For the device itself, the channel ID is 0. For channels under the device, the channel IDs are queried by the SDK interface.

**See also:**

[NETDEV\_InputVoiceData](#_语音数据转发), [NETDEV\_SetPlayDecodeAudioCB](#_注册解码后音频数据回调), [NETDEV\_StopInputVoiceSrv](#_停止语音数据转发)

### Audio data forwarding

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_InputVoiceData  (  LPVOID lpVoiceComHandle,  LPVOID lpDataBuf,  INT32 dwDataLen,  [LPNETDEV\_AUDIO\_SAMPLE\_PARAM\_S](#_音频参数结构体) pstVoiceParam  ); |

**Interface description:**

Enable audio data forwarding

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpVoiceComHandle | IN | User login handle, which is the return value from the [NETDEV\_StartInputVoiceSrv](#_开启语音数据转发) interface |
| lpDataBuf | IN | PCM audio data |
| dwDataLen | IN | Length of audio data |
| pstVoiceParam | IN | Audio parameter |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_StartInputVoiceSrv](#_开启语音数据转发), [NETDEV\_SetPlayDecodeAudioCB](#_注册解码后音频数据回调), [NETDEV\_StopInputVoiceSrv](#_停止语音数据转发)

### Stop audio data forwarding

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopInputVoiceSrv (LPVOID lpVoiceComHandle); |

**Interface description:**

Stop two-way audio

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpVoiceComHandle | IN | Turn on audio data forwarding handle, which is the return value from [NETDEV\_StartInputVoiceSrv](#_开启语音数据转发). |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_StopInputVoiceSrv](#_停止语音数据转发)

### Start audio broadcast

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_StartVoiceBroadcast  (  LPVOID lpUserID,  INT32 dwChannelID  ); |

**Interface description:**

Start audio broadcast

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwChannelID | IN | Channel ID |

**Return value:**

0 means failure, other values are used as handle of functions such as [NETDEV\_StopVoiceBroadcast](#_停止语音广播). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* For the device itself, the channel ID is 0. For channels under the device, the channel IDs are queried by the SDK interface.

**See also:**

[NETDEV\_StopVoiceBroadcast](#_停止语音广播)

### Stop audio broadcast

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopVoiceBroadcast (LPVOID lpPlayHandle); |

**Interface description:**

Stop audio broadcast

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Audio broadcast handle,  which is the return value from the [NETDEV\_StartVoiceBroadcast](#_开启语音广播) and [NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组) interfaces. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_StartVoiceBroadcast](#_开启语音广播), [NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)

### Enable audio broadcast group

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_CreateVoiceBroadcastGroup  (  LPVOID lpUserID,  [LPNETDEV\_OPERATE\_LIST\_S](#_批处理列表) pstChnList  )； |

**Interface description:**

Enable audio broadcast group

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstChnList | IN | Channel list |

**Return value:**

0 means failure, other values are used as handle of functions such as [NETDEV\_StopVoiceBroadcast](#_停止语音广播). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Supported by VMS only.

**See also:**

[NETDEV\_ModifyVoiceBroadcastGroup](#_修改语音广播组)

### Modify audio broadcast group

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyVoiceBroadcastGroup  (  LPVOID lpPlayHandle,  [LPNETDEV\_MODIFY\_VOICE\_BROADCAST\_INFO\_S](#_语音广播组信息结构体) pstModifyInfo  ); |

**Interface description:**

Modify audio broadcast group

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Audio broadcast group handle, which is the return value from the [NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组) interface |
| pstModifyInfo | IN | Modify audio broadcast group info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Supported by VMS only.

**See also:**

[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)

### Get channel status in audio broadcast group

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetVoiceBroadcastChlStatus  (  LPVOID lpPlayHandle,  [LPNETDEV\_VOICE\_BROADCAST\_GROUP\_INFO\_S](#_语音广播组通道信息结构体) pstGroupInfo  ); |

**Interface description:**

Get channel status in audio broadcast group

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Audio broadcast group handle, which is the return value from the [NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组) interface |
| pstGroupInfo | OUT | Audio broadcast group info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Supported by VMS only.

**See also:**

[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)

### Modify channel status in audio broadcast group

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyVoiceBroadcastStatus  (  LPVOID lpPlayHandle,  [LPNETDEV\_VOICE\_BROADCAST\_CTRL\_INFO\_S](#_语音广播组控制信息结构体) pstCtrlInfo  ); |

**Interface description:**

Modify channel status in audio broadcast group

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Audio broadcast group handle, which is the return value from the [NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组) interface |
| pstCtrlInfo | IN | Audio broadcast group control info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Supported by VMS only.

**See also:**

[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)

### Turn on speaker sound

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_OpenSound(IN LPVOID lpPlayHandle); |

**Interface description:**

Turn on speaker sound

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |

Remarks:

* Stream startup handle, which can be the return value from the [NETDEV\_RealPlay](#_实时预览), [NETDEV\_StartVoiceCom](#_开启语音对讲) interfaces.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SoundVolumeControl](#_设置扬声器音量), [NETDEV\_GetSoundVolume](#_获取扬声器音量), [NETDEV\_CloseSound](#_关闭扬声器声音)

### Get speaker volume

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetSoundVolume  (  LPVOID lpPlayHandle,  INT32\* pdwVolume  ); |

**Interface description:**

Get speaker volume

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |
| pdwVolume | OUT | Sound volume. |

Remarks:

* Stream startup handle, which can be the return value from the [NETDEV\_RealPlay](#_实时预览), [NETDEV\_StartVoiceCom](#_开启语音对讲) interfaces.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_OpenSound](#_开启扬声器声音), [NETDEV\_SoundVolumeControl](#_设置扬声器音量), [NETDEV\_CloseSound](#_关闭扬声器声音)

### Set speaker volume

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SoundVolumeControl  (  LPVOID lpPlayHandle,  INT32 dwVolume  ); |

**Interface description:**

Set speaker volume

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |
| dwVolume | IN | Sound volume. Range: [0-255] |

Remarks:

* Stream startup handle, which can be the return value from the [NETDEV\_RealPlay](#_实时预览), [NETDEV\_StartVoiceCom](#_开启语音对讲) interfaces.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_OpenSound](#_开启扬声器声音), [NETDEV\_GetSoundVolume](#_获取扬声器音量), [NETDEV\_CloseSound](#_关闭扬声器声音)

### Turn off speaker sound

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_CloseSound(LPVOID lpPlayHandle) |

**Interface description:**

Turn off speaker sound

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |

Remarks:

* Stream startup handle, which can be the return value from the [NETDEV\_RealPlay](#_实时预览), [NETDEV\_StartVoiceCom](#_开启语音对讲) interfaces.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_OpenSound](#_开启扬声器声音), [NETDEV\_SoundVolumeControl](#_设置扬声器音量), [NETDEV\_GetSoundVolume](#_获取扬声器音量)

### Set mute status

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetMuteStatus  (  LPVOID lpPlayHandle,  BOOL bMute  ); |

**Interface description:**

Set mute status

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |
| bMute | IN | Mute status. 0: Disable mute 1: Enable mute |

Remarks:

* Stream startup handle, which can be the return value from the [NETDEV\_RealPlay](#_实时预览), [NETDEV\_StartVoiceCom](#_开启语音对讲) interfaces.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetMuteStatus](#_获取静音状态)

### Get mute status

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetMuteStatus  (  LPVOID lpPlayHandle,  BOOL\* pbMute  ); |

**Interface description:**

Get mute status

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |
| pbMute | OUT | Mute status. 0: Disable mute 1: Enable mute |

Remarks:

* Stream startup handle, which can be the return value from the [NETDEV\_RealPlay](#_实时预览), [NETDEV\_StartVoiceCom](#_开启语音对讲) interfaces.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SetMuteStatus](#_设置静音状态)

### Turn on microphone

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_OpenMic (LPVOID lpPlayHandle); |

**Interface description:**

Turn on microphone

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |

Remarks:

* Stream startup handle, which can be the return value from the [NETDEV\_StartVoiceCom](#_开启语音对讲), [NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组) interfaces.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_MicVolumeControl](#_设置麦克风音量), [NETDEV\_GetMicVolume](#_获取麦克风音量), [NETDEV\_CloseMic](#_关闭麦克风)

### Get microphone volume

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetMicVolume  (  LPVOID lpPlayHandle，  INT32\* pdwVolume  ); |

**Interface description:**

Get microphone volume

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |
| pdwVolume | OUT | Sound volume. Range: [0-255] |

Remarks:

* Stream startup handle, which can be the return value from the [NETDEV\_StartVoiceCom](#_开启语音对讲), [NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组) interfaces.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_OpenMic](#_开启麦克风), [NETDEV\_MicVolumeControl](#_设置麦克风音量), [NETDEV\_CloseMic](#_关闭麦克风)

### Set microphone volume

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_MicVolumeControl  (  LPVOID lpPlayHandle，  INT32 dwVolume  ); |

**Interface description:**

Set microphone volume

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |
| dwVolume | OUT | Sound volume. Range: [0-255] |

Remarks:

* Stream startup handle, which can be the return value from the [NETDEV\_StartVoiceCom](#_开启语音对讲), [NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组) interfaces.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_OpenMic](#_开启麦克风), [NETDEV\_GetMicVolume](#_获取麦克风音量), [NETDEV\_CloseMic](#_关闭麦克风)

### Turn off microphone

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_CloseMic (IN LPVOID lpPlayHandle); |

**Interface description:**

Turn off microphone

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Stream startup handle |

Remarks:

* Stream startup handle, which can be the return value from the [NETDEV\_StartVoiceCom](#_开启语音对讲), [NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组) interfaces.

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_OpenMic](#_开启麦克风), [NETDEV\_MicVolumeControl](#_设置麦克风音量), [NETDEV\_GetMicVolume](#_获取麦克风音量)

## Play Local Video

### Open local file

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_OpenMediaFile(CHAR \*pszFilename); |

**Interface description:**

Open local file

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| pszFilename | IN | Filename, including the absolute path. The recommended length is no more than 256 bytes. |

**Return value:**

0 means failure, other values are used as handle of functions such as [NETDEV\_StartPlayMediaFile](#_开始播放本地文件). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_StartPlayMediaFile](#_开始播放本地文件), [NETDEV\_GetMediaFileTime](#_获取本地文件播放总时长), [NETDEV\_StopPlayMediaFile](#_停止播放本地文件)

### Start to play local file

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StartPlayMediaFile  (  LPVOID lpPlayHandle,  LPVOID lpPlayWnd  ); |

**Interface description:**

Start to play local file

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | File handle, which is the return value from the [NETDEV\_OpenMediaFile](#_打开本地文件) interface |
| lpPlayWnd | IN | Window handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Call [NETDEV\_PlayBackControl](#_回放控制) for playback control.

**See also:**

[NETDEV\_OpenMediaFile](#_打开本地文件), [NETDEV\_GetMediaFileTime](#_获取本地文件播放总时长), [NETDEV\_StopPlayMediaFile](#_停止播放本地文件)

### Stop playing local file

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopPlayMediaFile(LPVOID lpPlayHandle); |

**Interface description:**

Stop playing local file

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | File handle, which is the return value from the [NETDEV\_OpenMediaFile](#_打开本地文件) interface |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_OpenMediaFile](#_打开本地文件), [NETDEV\_StartPlayMediaFile](#_开始播放本地文件), [NETDEV\_GetMediaFileTime](#_获取本地文件播放总时长)

### Get total duration of local file

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetMediaFileTime  (  LPVOID lpPlayHandle,  INT32\* pdwTotalTime  ); |

**Interface description:**

Stop playing local file

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | File handle, which is the return value from the [NETDEV\_OpenMediaFile](#_打开本地文件) interface |
| pdwTotalTime | OUT | Total duration of local file (s) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_OpenMediaFile](#_打开本地文件), [NETDEV\_StartPlayMediaFile](#_开始播放本地文件), [NETDEV\_StopPlayMediaFile](#_停止播放本地文件)

## Fisheye Dewarping

### Determine fisheye stream

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_IsFishEyeStream  (  LPVOID lpPlayHandle,  BOOL \*pbFishEyeStream  ) |

**Interface description:**

Determine whether is a fisheye stream

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle, return value from the [NETDEV\_RealPlay](#_实时预览) interface |
| pbFishEyeStream | OUT | Whether is a fisheye stream |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Set 5ePTZ display mode

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPtzDisplayMode  (  LPVOID lpPlayHandle,  UINT32 udwPtzMode  ); |

**Interface description:**

Set 5ePTZ display mode

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle, return value from the [NETDEV\_RealPlay](#_实时预览) interface |
| udwPtzMode | IN | 5ePTZ display mode. See [NETDEV\_PTZ\_DISPLAY\_MODE](#_5ePTZ显示模式枚举定义) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Mouse movement mode

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetMouseMoveMode  (  LPVOID lpPlayHandle,  INT32 dwOperateMode,  UINT32 udwFlag,  INT16 wDelta,  [LPNETDEV\_POINT\_S](#_点坐标结构体) pstPoint  ); |

**Interface description:**

Mouse movement mode

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle, return value from the [NETDEV\_RealPlay](#_实时预览) interface |
| dwOperateMode | IN | Mouse operation mode. See [NETDEV\_MOUSE\_MOVE\_MODE\_E](#_鼠标移动模式枚举) |
| udwFlag | IN | Reserved |
| wDelta | IN | Mouse wheel offset (optional) |
| pstPoint | IN | Movement coordinates |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get fisheye dewarping parameters

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetFishEyeParam  (  LPVOID lpPlayHandle,  [LPNETDEV\_POINT\_S](#_点坐标结构体) pstCenterPoint,  UINT32 \*pudwRadius,  LPVOID lpParm  ); |

**Interface description:**

Get fisheye dewarping parameters

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle, return value from the [NETDEV\_RealPlay](#_实时预览) interface |
| pstCenterPoint | OUT | Center point of circular fisheye image |
| pudwRadius | OUT | Radius of circular fisheye image |
| lpParm | OUT | Reserved fisheye lens parameter |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Set mode and mount position

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPtzAndFixMode  (  LPVOID lpPlayHandle,  INT32 dwPtzMode,  INT32 dwInstallMode  ) |

**Interface description:**

Set mode and mount position

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle, return value from the [NETDEV\_RealPlay](#_实时预览) interface |
| dwPtzMode | IN | Fisheye PTZ mode. See [NETDEV\_FISHEYE\_PTZ\_MODE\_E](#_鱼眼设备矫正模式枚举) |
| dwInstallMode | IN | Installation position. See [NETDEV\_INSTALL\_MODE\_E](#_设备安装模式枚举定义) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetPtzAndFixMode](#_获取模式和安装位置)

### Get mode and mount position

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetPtzAndFixMode  (  LPVOID lpPlayHandle,  INT32 dwPtzMode,  INT32 dwInstallMode  ) |

**Interface description:**

Mouse movement mode

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle, return value from the [NETDEV\_RealPlay](#_实时预览) interface |
| dwPtzMode | OUT | Fisheye PTZ mode. See [NETDEV\_FISHEYE\_PTZ\_MODE\_E](#_鱼眼设备矫正模式枚举) |
| dwInstallMode | OUT | Installation position. See [NETDEV\_INSTALL\_MODE\_E](#_设备安装模式枚举) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SetPtzAndFixMode](#_设置模式和安装位置)

## Manual Recording

### Get manual recording status

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetManualRecordStatus  (  LPVOID lpUserID,  [LPNETDEV\_MANUAL\_RECORD\_CFG\_S](#_手动录像结构体) pstManualRecordCfg,  UINT32 \*pudwRecodeStatus  ); |

**Interface description:**

Get manual recording status

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstManualRecordCfg | IN | Manual recording parameter info |
| pudwRecodeStatus | OUT | Manual recording status. See [NETDEV\_MANUAL\_RECORD\_STATUS\_E](#_录像状态枚举_1) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Start manual recording

|  |
| --- |
| BOOL STDCALL NETDEV\_StartManualRecord  (  LPVOID lpUserID,  [LPNETDEV\_MANUAL\_RECORD\_CFG\_S](#_手动录像结构体) pstManualRecordCfg  ); |

**Interface description:**

Start manual recording

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstManualRecordCfg | IN | Manual recording parameter info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_StopManualRecord](#_停止手动录像)

### Stop manual recording

|  |
| --- |
| BOOL STDCALL NETDEV\_StopManualRecord  (  LPVOID lpUserID,  [LPNETDEV\_MANUAL\_RECORD\_CFG\_S](#_手动录像结构体) pstManualRecordCfg  ); |

**Interface description:**

Stop manual recording

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstManualRecordCfg | IN | Manual recording parameter info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_StartManualRecord](#_开启手动录像)

## PTZ Control

### 3D positioning

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZ3DPosition  (  LPVOID lpPlayHandle,  [LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S](#_拉框缩放结构体) pstZoomAreaInfo  ); |

**Interface description:**

3D positioning (preview needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle |
| pstZoomAreaInfo | IN | Structure of drag-to-zoom |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZ3DPosition\_Other](#_云台3D定位)

### 3D positioning without preview

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZ3DPosition\_Other  (  LPVOID                                                         lpUserID,  INT32                                                            dwChannelID,  [LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S](#_拉框缩放结构体)    pstZoomAreaInfo  ); |

**Interface description:**

3D positioning (preview not needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChannelID | IN | Channel ID |
| pstZoomAreaInfo | IN | Structure of drag-to-zoom |

Remarks:

* No need to start live preview

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZ3DPosition](#_云台3D定位_1)

### Drag to zoom in

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZAreaZoomIn  (  LPVOID lpPlayHandle,  [LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S](#_拉框缩放结构体) pstZoomAreaInfo  )； |

**Interface description:**

Drag to zoom in (preview needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle |
| pstZoomAreaInfo | IN | Structure of drag-to-zoom |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZAreaZoomIn\_Other](#_云台拉框放大)

### Drag to zoom in without preview

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZAreaZoomIn\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S](#_拉框缩放结构体) pstZoomAreaInfo  ); |

**Interface description:**

Drag to zoom in (preview not needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChannelID | IN | Channel ID |
| pstZoomAreaInfo | IN | Structure of drag-to-zoom |

Remarks:

* No need to start live preview

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZAreaZoomIn](#_云台拉框放大_1)

### Drag to zoom out

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZAreaZoomOut  (  LPVOID lpPlayHandle,  [LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S](#_拉框缩放结构体) pstZoomAreaInfo  ); |

**Interface description:**

Drag to zoom out (preview not needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle |
| pstZoomAreaInfo | IN | Structure of drag-to-zoom |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZAreaZoomOut\_Other](#_拉框放大.缩小)

### Drag to zoom out without preview

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZAreaZoomOut\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S](#_拉框缩放结构体) pstZoomAreaInfo  ); |

**Interface description:**

Drag to zoom out (preview not needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChannelID | IN | Channel ID |
| pstZoomAreaInfo | IN | Structure of drag-to-zoom |

Remarks:

* No need to start live preview

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZAreaZoomOut](#_云台拉框缩小)

### Drag to zoom in/out

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZSelZoomIn  (  LPVOID lpPlayHandle,  [LPNETDEV\_PTZ\_OPERATEAREA\_S](#_拉框放大结构体_1) pstPtzOperateArea  ); |

**Interface description:**

Drag to zoom in/out (preview needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle |
| pstPtzOperateArea | IN | Info about structure of drag-to-zoom |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZSelZoomIn\_Other](#_拉框放大.缩小)

### Drag to zoom in/out without preview

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZSelZoomIn\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_OPERATEAREA\_S](#_拉框放大结构体_1) pstPtzOperateArea  ); |

**Interface description:**

Drag to zoom in/out (preview not needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| pstPtzOperateArea | IN | Image area info Pay attention to the parameter value range |

Remarks:

* No need to start live preview

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZSelZoomIn](#_拉框放大.缩小_1)

### PTZ control operations

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZControl  (  LPVOID lpPlayHandle,  INT32 dwPTZCommand,  INT32 dwSpeed  )； |

**Interface description:**

PTZ control operation (need to start live preview)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle |
| dwPTZCommand | IN | PTZ control command. See [NETDEV\_PTZ\_E](#_云台命令枚举) |
| dwSpeed | IN | PTZ control speed. Range: [1, 9] |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZControl\_Other](#_云台控制操作)

### PTZ control operation without preview

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZControl\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwPTZCommand,  INT32 dwSpeed  ); |

**Interface description:**

PTZ control operation (no need to start live preview)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| dwPTZCommand | IN | PTZ control command. See [NETDEV\_PTZ\_E](#_云台命令枚举) |
| dwSpeed | IN | PTZ control speed. Range: [1, 9] |

Remarks:

* No need to start live preview

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZControl](#_云台控制操作_1)

### Get or set auto guard info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZGetGuard\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwPTZGuardCmd,  [LPNETDEV\_PTZ\_GUARD\_INFO\_S](#_云台守望信息结构体) pstPTZGuardInfo  ); |

**Interface description:**

Get or set auto guard info of specified channel (preview not needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChannelID | IN | Channel ID |
| dwPTZGuardCmd | IN | PTZ auto guard configuration commands See [NETDEV\_PTZ\_GUARD\_CMD\_E](#_云台守望配置命令) |
| pstPTZGuardInfo | OUT | Structure of drag-to-zoom |

Remarks:

* No need to start live preview

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZHomePosition](#_云台看守位操作), [NETDEV\_PTZHomePosition\_Other](#_云台看守位操作_1)

### Home position operation

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZHomePosition  (  LPVOID lpRealHandle,  INT32 dwPTZHomePositionCmd  ); |

**Interface description:**

Home position operation (preview needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpRealHandle | IN | Live view handle |
| dwPTZHomePositionCmd | IN | Home position operation commands. See  [NETDEV\_PTZ\_HOMEPOSITIONCMD\_E](#_云台看守位操作命令) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZHomePosition\_Other](#_云台看守位操作_1)

### Home position operation without preview

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZHomePosition\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwPTZHomePositionCmd  ); |

**Interface description:**

Home position operation (preview not needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| dwPTZHomePositionCmd | IN | Home position operation commands See [NETDEV\_PTZ\_HOMEPOSITIONCMD\_E](#_云台看守位操作命令) |

Remarks:

* No need to start live preview

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZHomePosition](#_云台看守位操作)

### Absolute PTZ coordinates movement

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZAbsoluteMove  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_ABSOLUTE\_MOVE\_S](#_绝对坐标移动) pstAbsoluteMove  ); |

**Interface description:**

Absolute PTZ coordinates movement

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| pstAbsoluteMove | IN | Structure of absolute PTZ coordinates movement |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

### Get PTZ status

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZGetStatus  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_STATUS\_S](#_云台状态) pstPTZStaus  ); |

**Interface description:**

Get PTZ status

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| pstPTZStaus | OUT | PTZ status |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get recorded patrol route

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZGetTrackCruise  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_TRACK\_INFO\_S](#_云台轨迹巡航路径结构体) pstTrackCruiseInfo  ); |

**Interface description:**

Get recorded patrol route

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| pstTrackCruiseInfo | OUT | Patrol route list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZGetTrackCruise\_V30](#_获取云台轨迹巡航路径)

### Get recorded patrol route (including patrol route ID)

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZGetTrackCruise\_V30  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_TRACK\_LIST\_V30\_S](#_云台轨迹巡航路径列表) pstTrackCruiseList  ); |

**Interface description:**

Get recorded patrol route

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChannelID | IN | Channel ID |
| pstTrackCruiseList | OUT | Patrol route list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZGetTrackCruise](#_获取云台轨迹巡航路径_1)

### Get patrol status of specified channel

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZGetTrackStatus  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_TRACK\_STATUS\_INFO\_S](#_云台轨迹巡航状态信息结构体) pstTrackStatus  ); |

**Interface description:**

Get patrol status of specified channel

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChannelID | IN | Channel ID |
| pstTrackStatus | OUT | Structure of patrol status info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZTrackCruise](#_获取云台轨迹巡航路径_1)， [NETDEV\_PTZTrackCruise\_V30](#_获取云台轨迹巡航路径)

### Recorded patrol operation

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZTrackCruise  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwPTZTrackCruiseCmd,  CHAR \*pszTrackCruiseName  ); |

**Interface description:**

Recorded patrol operation (preview not needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| dwPTZTrackCruiseCmd | IN | Recorded patrol operation commands See [NETDEV\_PTZ\_TRACKCMD\_E](#_云台巡航操作枚举) |
| pszTrackCruiseName | INOUT | Name of recorded patrol. Recommended length: 64 bytes. User allocates RAM. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZTrackCruise\_V30](#_云台轨迹巡航操作)

### Recorded patrol operation without preview

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZTrackCruise\_V30  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwPTZTrackCruiseCmd,  [LPNETDEV\_PTZ\_TRACK\_INFO\_V30\_S](#_云台轨迹巡航路径结构体_1) pstTrackCruiseInfo  ); |

**Interface description:**

Recorded patrol operation (preview not needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChannelID | IN | Channel ID |
| dwPTZTrackCruiseCmd | IN | Recorded patrol operation commands See [NETDEV\_PTZ\_TRACKCMD\_E](#_云台巡航操作枚举) |
| pstTrackCruiseInfo | IN | Recorded patrol route info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZTrackCruise](#_云台轨迹巡航操作_1)

### Get preset position list

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetPTZPresetList  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_ALLPRESETS\_S](#_所有云台预置位结构体) pstPresetList  ); |

**Interface description:**

Get preset position list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChannelID | IN | Channel ID |
| pstPresetList | OUT | Preset list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZPreset](#_云台预置位操作)，[NETDEV\_PTZPreset\_Other](#_云台预置位操作_1)

### Preset operation

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZPreset  (  LPVOID lpPlayHandle,  INT32 dwPTZPresetCmd,  CHAR \*pszPresetName,  INT32 dwPresetID  ); |

**Interface description:**

Preset operation (preview needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle |
| dwPTZPresetCmd | IN | Preset operation commands See [NETDEV\_PTZ\_PRESETCMD\_E](#_云台预置位操作命令枚举) |
| pszPresetName | IN | Preset name |
| dwPresetID | IN | Preset ID (starts from 1). Up to 255 presets are allowed. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZPreset\_Other](#_云台预置位操作_1)

### Preset operation without preview

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZPreset\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwPTZPresetCmd,  CHAR \*pszPresetName,  INT32 dwPresetID  ); |

**Interface description:**

Preset operation (preview not needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| dwPTZPresetCmd | IN | Preset operation commands See [NETDEV\_PTZ\_PRESETCMD\_E](#_云台预置位操作命令枚举) |
| pszPresetName | IN | Preset name |
| dwPresetID | IN | Preset ID (starts from 1). Up to 255 presets are allowed. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZPreset](#_云台预置位操作)

### Preset patrol operation

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZCruise  (  LPVOID lpPlayHandle,  INT32 dwPTZCruiseCmd,  [LPNETDEV\_CRUISE\_INFO\_S](#_云台预置位巡航路径详细信息结构体) pstCruiseInfo  ); |

**Interface description:**

Preset patrol operation (preview needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Live view handle |
| dwPTZCruiseCmd | IN | Patrol operation commands See [NETDEV\_PTZ\_CRUISECMD\_E](#_云台巡航操作枚举_1) |
| pstCruiseInfo | IN | Patrol route info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZCruise\_Other](#_云台预置位巡航操作)

### Preset patrol operation without preview

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZCruise\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwPTZCruiseCmd,  [LPNETDEV\_CRUISE\_INFO\_S](#_云台预置位巡航路径详细信息结构体) pstCruiseInfo  ); |

**Interface description:**

Preset patrol operation (preview not needed)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| dwPTZCruiseCmd | IN | Patrol operation commands See [NETDEV\_PTZ\_CRUISECMD\_E](#_云台巡航操作枚举_1) |
| pstCruiseInfo | IN | Patrol route info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZCruise](#_云台预置位巡航操作_1)

### Get preset patrol route

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZGetCruise  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_CRUISE\_LIST\_S](#_云台预置位巡航路径列表结构体) pstCruiseList  ); |

**Interface description:**

Get preset patrol route

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| pstCruiseList | OUT | Patrol route list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_PTZCruise](#_云台预置位巡航操作_1), [NETDEV\_PTZCruise\_Other](#_云台预置位巡航操作)

### PTZ calibration

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZCalibrate  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_ORIENTATION\_INFO\_S](#_云台方位标定信息结构体) pstOrientationInfo  ); |

**Interface description:**

PTZ calibration

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| pstOrientationInfo | IN | PTZ direction info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get current lens angle

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZGetLensAngle  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_LENSANGLE\_S](#_云台镜头当前视场角结构体) pstLensAngleInfo  ); |

**Interface description:**

Get current lens angle of specified video channel

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChannelID | IN | Channel ID |
| pstLensAngleInfo | OUT | Current lens angle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get PTZ latitude and longitude info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetPTZAbsolutePTInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_PT\_POSITION\_INFO\_S](#_云台经纬度信息单位) pstPTPositionInfo  ); |

**Interface description:**

Get PTZ latitude and longitude info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| pstPTPositionInfo | OUT | Position info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SetPTZAbsolutePTInfo](#_设置云台的经纬度信息)

### Set PTZ latitude and longitude info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPTZAbsolutePTInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_PT\_POSITION\_INFO\_S](#_云台经纬度信息单位) pstPTPositionInfo  ); |

**Interface description:**

Set PTZ latitude and longitude info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| pstPTPositionInfo | IN | Position info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetPTZAbsolutePTInfo](#_获取云台的经纬度信息)

### Get current zoom ratio

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetPTZAbsoluteZoomInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  FLOAT \*pfZoomRatio  ); |

**Interface description:**

Get current zoom ratio

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| pfZoomRatio | OUT | Position info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SetPTZAbsoluteZoomInfo](#_设置云台当前变倍倍数)

### Set zoom ratio

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPTZAbsoluteZoomInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  FLOAT fZoomRatio  ); |

**Interface description:**

Set zoom ratio

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| fZoomRatio | IN | Position info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetPTZAbsoluteZoomInfo](#_获取云台当前变倍倍数)

## System Configuration

### Enable/disable Telnet

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_EnableTelnet  (  LPVOID lpUserID,  BOOL bEnable  ); |

**Interface description:**

Enable/disable Telnet

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| bEnable | IN | Telnet status. 0: Disable 1: Enable |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Export configuration file

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetConfigFile  (  LPVOID lpUserID,  CHAR\* pszConfigPath  ); |

**Interface description:**

Export configuration file

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| pszConfigPath | IN | Configuration file path (including filename, suffixed with tgz) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SetConfigFile](#_设置配置文件业务)

### Import configuration file

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetConfigFile  (  LPVOID lpUserID,  CHAR\* pszConfigPath  ); |

**Interface description:**

Import configuration file

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| pszConfigPath | IN | Configuration file path (including filename, format: device model\_IP\_config.tgz, example: HIC5621E-L-U\_192.168.3.112\_config.tgz) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetConfigFile](#_导出配置文件业务)

### Get device configuration info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDevConfig  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwCommand,  LPVOID lpOutBuffer,  INT32 dwOutBufferSize,  INT32 \*pdwBytesReturned  ); |

**Interface description:**

Get device configuration info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| dwCommand | IN | Device configuration commands See [NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令) |
| lpOutBuffer | INOUT | Pointer to data receiving buffer |
| dwOutBufferSize | OUT | Size of data receiving buffer (unit: byte), cannot be 0. |
| pdwBytesReturned | OUT | Pointer to the length of received data, cannot be NULL. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SetDevConfig](#_设置设备的配置信息)

### Set device configuration info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetDevConfig  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwCommand,  LPVOID lpInBuffer,  INT32 dwInBufferSize  ); |

**Interface description:**

Set device configuration info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| dwCommand | IN | Device configuration commands See [NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令) |
| lpInBuffer | IN | Pointer to the data receiving buffer |
| dwInBufferSize | IN | Size of data receiving buffer (unit: byte) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetDevConfig](#_获取设备的配置信息)

### Get disk mode

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDiskMode  (  LPVOID lpUserID,  UINT32 \* pudwDiskMode  )； |

**Interface description:**

Get disk mode

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| pudwDiskMode | OUT | Disk mode See [NETDEV\_DISK\_MODE\_TYPE\_E](#_硬盘模式) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SetDiskMode](#_设置硬盘模式)

### Set disk mode

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetDiskMode  (  LPVOID lpUserID,  UINT32 udwDiskMode  ); |

**Interface description:**

Set disk mode

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| udwDiskMode | IN | Disk mode See [NETDEV\_DISK\_MODE\_TYPE\_E](#_硬盘模式) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetDiskMode](#_查询硬盘模式)

### Get system time of device

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetSystemTimeCfg  (  LPVOID lpUserID,  [LPNETDEV\_TIME\_CFG\_S](#_时间配置_3) pstSystemTimeInfo  ); |

**Interface description:**

Get system time of device

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstSystemTimeInfo | OUT | Time configuration info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SetSystemTimeCfg](#_设置设备系统时间配置)

### Set system time of device

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetSystemTimeCfg  (  LPVOID lpUserID,  [LPNETDEV\_TIME\_CFG\_S](#_时间配置_3) pstSystemTimeInfo  ); |

**Interface description:**

Set system time of device

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstSystemTimeInfo | IN | Time configuration info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetSystemTimeCfg](#_获取设备系统时间配置)

### Modify device IP address without login

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyDeviceAddr  (  [LPNETDEV\_DEV\_ADDR\_INFO\_S](#_设备地址信息结构体) pstDevAddrInfo  ) |

**Interface description:**

Modify device IP address (without login)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| pstDevAddrInfo | IN | Address of the device to be modified |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Change device name

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyDeviceName  (  LPVOID lpUserID,  CHAR \*pszDeviceName  ); |

**Interface description:**

Change device name

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pszDeviceName | IN | Device name |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get device capabilities

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDeviceCapability  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwCommand,  LPVOID lpOutBuffer,  INT32 dwOutBufferSize,  INT32 \*pdwBytesReturned  ); |

**Interface description:**

Get device capabilities

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| dwCommand | IN | Device capability type command See [NETDEV\_CAPABILITY\_COMMOND\_E](#_能力集命令) |
| lpOutBuffer | OUT | Pointer to data receiving buffer |
| dwOutBufferSize | OUT | Size of data receiving buffer (unit: byte) |
| pdwBytesReturned | OUT | Pointer to the length of the received data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get device list by device type

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindDevList  (  LPVOID lpUserID,  INT32 dwDevType  ); |

**Interface description:**

Get device list by device type

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwDevType | IN | Device type. See [NETDEV\_DEVICE\_MAIN\_TYPE\_E](#_设备类型枚举定义) |

**Return value:**

Search handle. 0 means failure, other values are used as parameters of functions such as [NETDEV\_FindNextDevInfo](#_逐个获取查找到的设备信息), [NETDEV\_FindCloseDevInfo](#_关闭查找设备信息，释放资源).

**See also:**

[NETDEV\_FindNextDevInfo](#_逐个获取查找到的设备信息), [NETDEV\_FindCloseDevInfo](#_关闭查找设备信息，释放资源)

### Get device info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextDevInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1) pstDevBasicInfo  ); |

**Interface description:**

Get device list by device type

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| pstDevBasicInfo | OUT | Pointer to the saved basic device info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_FindDevList](#_通过设备类型查询设备列表), [NETDEV\_FindCloseDevInfo](#_关闭查找设备信息，释放资源)

### Stop searching device info, release resource

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseDevInfo(LPVOID lpFindHandle); |

**Interface description:**

Stop searching device info, release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | File search handle |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

[NETDEV\_FindDevList](#_通过设备类型查询设备列表), [NETDEV\_FindNextDevInfo](#_逐个获取查找到的设备信息)

### Get device channel list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindDevChlBasicInfoList(LPVOID lpUserID); |

**Interface description:**

Get device channel list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |

**Return value:**

Service ID. 0 means failure, other values are used as parameters of functions such as [NETDEV\_FindNextChlDeviceInfo](#_逐个获取通道设备基本信息), [NETDEV\_FindCloseDevChlBasicInfo](#_关闭获取通道设备基本信息).

**See also:**

[NETDEV\_FindNextChlDeviceInfo](#_逐个获取通道设备基本信息), [NETDEV\_FindCloseDevChlBasicInfo](#_关闭获取通道设备基本信息)

### Get device channel basic info one bye one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextDevChlBasicInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_DEVICE\_CHL\_INFO\_S](#_设备通道基本信息) pstDeviceInfo  ); |

**Interface description:**

Get device channel basic info one bye one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| pstDeviceInfo | OUT | Pointer to the device info structure |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

[NETDEV\_FindDevChlBasicInfoList](#_获取通道设备基本信息列表), [NETDEV\_FindCloseDevChlBasicInfo](#_关闭获取通道设备基本信息)

### Stop getting basic info about device channel

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseDevChlBasicInfo(IN LPVOID lpFindHandle); |

**Interface description:**

Stop getting basic info about device channel

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

[NETDEV\_FindDevChlBasicInfoList](#_获取通道设备基本信息列表), [NETDEV\_FindNextChlDeviceInfo](#_逐个获取通道设备基本信息)

### Get channel list by device ID or channel type

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindDevChnList  (  LPVOID lpUserID,  INT32 dwDevID,  INT32 dwChnType  ); |

**Interface description:**

Get channel list by device ID or channel type

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwDevID | IN | Device ID |
| dwChnType | IN | Channel type. See [NETDEV\_CHN\_TYPE\_E](#_通道类型) |

**Return value:**

Search handle. 0 means failure, other values are used as parameters of functions such as [NETDEV\_FindNextDevChn](#_逐个获取查找到的设备通道信息), [NETDEV\_FindCloseDevChn](#_关闭查找设备通道信息，释放资源).

**See also:**

[NETDEV\_FindNextDevChn](#_逐个获取查找到的设备通道信息), [NETDEV\_FindCloseDevChn](#_关闭查找设备通道信息，释放资源)

### Get device channel info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextDevChn  (  LPVOID lpFindHandle,  LPVOID lpOutBuffer,  INT32 dwOutBufferSize,  INT32 \*pdwBytesReturned  ); |

**Interface description:**

Get device channel info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| lpOutBuffer | OUT | Pointer to data receiving buffer |
| dwOutBufferSize | IN | Size of data receiving buffer (unit: byte), cannot be 0 |
| pdwBytesReturned | OUT | Pointer to the length of received data, cannot be NULL. |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

[NETDEV\_FindDevChnList](#_通过设备ID或通道类型查询通道信息列表), [NETDEV\_FindCloseDevChn](#_关闭查找设备通道信息，释放资源)

### Stop searching device channel info, release resource

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseDevChn(LPVOID lpFindHandle); |

**Interface description:**

Stop searching device channel info, release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

[NETDEV\_FindNextDevChn](#_逐个获取查找到的设备通道信息), [NETDEV\_FindDevChnList](#_通过设备ID或通道类型查询通道信息列表)

### Get detailed channel info by channel type and channel ID

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetChnDetailByChnType  (  LPVOID lpUserID,  INT32 dwChnID,  INT32 dwChnType,  LPVOID lpOutBuffer,  INT32 dwOutBufferSize,  INT32 \*pdwBytesReturned  ); |

**Interface description:**

Get detailed channel info by channel type and channel ID

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChnID | IN | Channel ID |
| dwChnType | IN | Channel type. See [NETDEV\_CHN\_TYPE\_E](#_通道类型) |
| lpOutBuffer | INOUT | Pointer to data receiving buffer |
| dwOutBufferSize | IN | Size of data receiving buffer (unit: byte), cannot be 0. |
| pdwBytesReturned | OUT | Pointer to the length of received data, cannot be NULL. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SetChnDetailByChnType](#_根据通道类型和通道ID设置报警输出通道的信息)

### Set alarm output channel info by channel type and channel ID

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetChnDetailByChnType  (  LPVOID lpUserID,  INT32 dwChnID,  INT32 dwChnType,  LPVOID lpOutBuffer,  INT32 dwOutBufferSize  ); |

**Interface description:**

Set alarm output channel info by channel type and channel ID

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChnID | IN | Channel ID |
| dwChnType | IN | Channel type. See [NETDEV\_CHN\_TYPE\_E](#_通道类型) |
| lpOutBuffer | IN | Pointer to data receiving buffer |
| dwOutBufferSize | IN | Size of data receiving buffer (unit: byte), cannot be 0. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetChnDetailByChnType](#_根据通道类型和通道ID获取通道详细信息)

### Get channel type

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetChnType  (  LPVOID lpUserID,  INT32 dwChnID,  INT32 \*pdwChnType  ); |

**Interface description:**

Get channel type

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChnID | IN | Channel ID |
| pdwChnType | OUT | Device channel type. See [NETDEV\_CHN\_TYPE\_E](#_通道类型) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get device channel basic info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDevChlBasicInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_DEVICE\_CHL\_INFO\_S](#_设备通道基本信息) pstDeviceChlInfo  ); |

**Interface description:**

Get device channel basic info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChannelID | IN | Channel ID |
| pstDeviceChlInfo | OUT | Basic device info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get basic device info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDeviceBaseInfo  (  LPVOID lpUserID,  [LPNETDEV\_DEVICE\_BASE\_INFO\_S](#_设备基础信息) pstDeviceInfo  ); |

**Interface description:**

Get device info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| pstDeviceInfo | OUT | Device info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get device info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDeviceInfo  (  LPVOID lpUserID ,  [LPNETDEV\_DEVICE\_INFO\_S](#_设备信息) pstDevInfo  ); |

**Interface description:**

Get device info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstDevInfo | OUT | Pointer to the device info structure |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get detailed device info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDeviceInfo\_V30  (  LPVOID lpUserID,  INT32 dwDevID,  [LPNETDEV\_DEV\_INFO\_V30\_S](#_设备详细信息) pstDevDetailInfo  ); |

**Interface description:**

Get detailed device info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwDevID | IN | Device ID |
| pstDevDetailInfo | OUT | Detailed device info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get device location info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetGeolocationInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_GEOLACATION\_INFO\_S](#_定位信息) pstGPSInfo  ); |

**Interface description:**

Get device location info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChannelID | IN | Channel ID |
| pstGPSInfo | OUT | Geolocation info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get Wi-Fi sniffer’s MAC address info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetWifiSnifferMacList  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_WIFISNIFFER\_MAC\_LIST\_S](#_WIFISnifferMac数组信息) pstMACList  ); |

**Interface description:**

Get Wi-Fi sniffer’s MAC address info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| dwChannelID | IN | Channel ID |
| pstMACList | OUT | Array of Wi-Fi sniffer’s MAC addresses |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get video channel list

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_QueryVideoChlDetailList  (  LPVOID lpUserID,  INT32 \*pdwChlCount,  [LPNETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_S](#_视频通道详细信息) pstVideoChlList  ); |

**Interface description:**

Get video channel list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pdwChlCount | INOUT | Number of channels |
| pdwChlCount | OUT | Channel capability list |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

[NETDEV\_QueryVideoChlDetailListEx](#_查询视频通道信息列表), [NETDEV\_QueryVideoChlInfo](#_查询视频通道信息)

### Get video channel list

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_QueryVideoChlDetailListEx  (  LPVOID lpUserID,  INT32 \*pdwChlCount,  [LPNETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S](#_视频通道详细信息_1) pstVideoChlList  ); |

**Interface description:**

Get video channel list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pdwChlCount | INOUT | Number of channels |
| pstVideoChlList | OUT | Channel capability list |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

[NETDEV\_QueryVideoChlDetailList](#_查询视频通道信息列表_1), [NETDEV\_QueryVideoChlInfo](#_查询视频通道信息)

### Get video channel info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_QueryVideoChlInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_S](#_视频通道详细信息) pstVideoChlInfo  ); |

**Interface description:**

Get video channel list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| pstVideoChlInfo | OUT | Video channel info |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

[NETDEV\_QueryVideoChlDetailList](#_查询视频通道信息列表_1), [NETDEV\_QueryVideoChlDetailListEx](#_查询视频通道信息列表)

### Import audio file

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ImportAudioFile  (  LPVOID lpUserID,  CHAR\* pszFilePath  ); |

**Interface description:**

Import audio file

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pszFilePath | IN | Audio file path. The audio file must be in PCM format. The file size must not exceed 100KB. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and determine the cause of error.

**See also:**

[NETDEV\_DeleteAudioFile](#_删除音频文件)

### Delete audio file

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteAudioFile  (  LPVOID lpUserID,  UINT32 udwID  ); |

**Interface description:**

Delete audio file

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwID | IN | Audio file ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and determine the cause of error.

**See also:**

[NETDEV\_ImportAudioFile](#_导入音频文件)

### Get audio file

#### Get all audio file info

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindAudioFileList  (  LPVOID lpUserID,  UINT32\* pudwNum  ); |

**Interface description:**

Get all audio file info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pudwNum | OUT | Number of audio files |

**Return value:**

Null means failure, other values are used as the input parameter of [NETDEV\_FindNextAudioFileInfo](#_逐个查询获取到的音频文件信息), [NETDEV\_FindCloseAudioFileList](#_关闭查找，释放资源_1). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* This interface is used with the [NETDEV\_FindNextAudioFileInfo](#_逐个查询获取到的音频文件信息) and [NETDEV\_FindCloseAudioFileList](#_关闭查找，释放资源_1) interfaces.
* After calling this interface successfully, call the [NETDEV\_FindNextAudioFileInfo](#_逐个查询获取到的音频文件信息) interface repeatedly to get the next info.
* The [NETDEV\_FindCloseAudioFileList](#_关闭查找，释放资源_1) interface must be called to release resource and stop searching after audio file info is obtained.

#### Get audio file info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextAudioFileInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_AUDIO\_FILE\_INFO\_S](#_音频文件信息) pstAudioFileInfo  ); |

**Interface description:**

Get audio file info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| pstAudioFileInfo | OUT | Audio file info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and determine the cause of error.

Remarks:

* This interface is used with the [NETDEV\_FindAudioFileList](#_获取所有的音频文件信息) and [NETDEV\_FindCloseAudioFileList](#_关闭查找，释放资源_1) interfaces.
* Call this interface repeatedly to get the next info.
* The [NETDEV\_FindCloseAudioFileList](#_关闭查找，释放资源_1) interface must be called to release resource and stop searching after audio file info is obtained.

#### Stop searching device channel info, release resource

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseAudioFileList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching audio file info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* This interface is used with the [NETDEV\_FindAudioFileList](#_获取所有的音频文件信息), [NETDEV\_FindNextAudioFileInfo](#_逐个查询获取到的音频文件信息) interfaces.
* The [NETDEV\_FindCloseAudioFileList](#_关闭查找，释放资源_1) interface must be called to release resource and stop searching after organization channel info is obtained.

### Get audio output parameters

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetAudioOutputCfg  (  LPVOID lpUserID,  UINT32 udwChannelID,  [LPNETDEV\_AUDIO\_OUTPUT\_CFG\_S](#_音频输出参数) pstAudioOutputCfg  ); |

**Interface description:**

Get audio output parameters

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwChannelID | IN | Channel ID |
| pstAudioOutputCfg | OUT | Audio output parameters |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and determine the cause of error.

### Set audio output parameters

|  |
| --- |
| BOOL STDCALL NETDEV\_SetAudioOutputCfg  (  LPVOID lpUserID,  UINT32 udwChannelID,  [LPNETDEV\_AUDIO\_OUTPUT\_CFG\_S](#_音频输出参数) pstAudioOutputCfg  ); |

**Interface description:**

Set audio output parameters

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwChannelID | IN | Channel ID |
| pstAudioOutputCfg | IN | Audio output parameters |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and determine the cause of error.

## Alarm-Triggered Snapshot

### Get alarm-triggered snapshot URL

#### Get alarm-triggered snapshot URL list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindAlarmSnapShotURL  (  LPVOID lpUserID,  [LPNETDEV\_ALARM\_SNAPSHOT\_COND\_S](#_告警抓图URL查找条件) pstAlarmSnapShotCond  ); |

**Interface description:**

Get alarm-triggered snapshot URL list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstAlarmSnapShotCond | IN | Condition for getting alarm snapshots |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextAlarmSnapShotURL](#_逐个查找获取到的URL信息) and [NETDEV\_FindCloseAlarmSnapShotURL](#_关闭告警联动抓图URL列表查找).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextAlarmSnapShotURL](#_逐个查找获取到的URL信息) and [NETDEV\_FindCloseAlarmSnapShotURL](#_关闭告警联动抓图URL列表查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextAlarmSnapShotURL](#_逐个查找获取到的URL信息) interface repeatedly to get the next URL.
* Call the [NETDEV\_FindCloseAlarmSnapShotURL](#_关闭告警联动抓图URL列表查找) interface to release resource and stop the search after alarm snapshot URL is obtained.

**See also:**

[NETDEV\_FindNextAlarmSnapShotURL](#_逐个查找获取到的URL信息), [NETDEV\_FindCloseAlarmSnapShotURL](#_关闭告警联动抓图URL列表查找)

#### Get alarm image info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextAlarmSnapShotURL  (  LPVOID lpFindHandle,  [LPNETDEV\_ALARM\_SNAPSHOT\_PIC\_S](#_告警抓拍图片信息) pstAlarmPicInfo  ); |

**Interface description:**

Get alarm snapshot info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_FindAlarmSnapShotURL](#_获取告警联动抓图URL列表) interface. |
| pstAlarmPicInfo | OUT | Alarm snapshot image info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindAlarmSnapShotURL](#_获取告警联动抓图URL列表) and [NETDEV\_FindCloseAlarmSnapShotURL](#_关闭告警联动抓图URL列表查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseAlarmSnapShotURL](#_关闭告警联动抓图URL列表查找) interface to release resource and stop the search after alarm snapshot URL is obtained.

**See also:**

[NETDEV\_FindAlarmSnapShotURL](#_获取告警联动抓图URL列表), [NETDEV\_FindCloseAlarmSnapShotURL](#_关闭告警联动抓图URL列表查找)

#### Stop searching alarm-triggered snapshot URL list

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseAlarmSnapShotURL  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching alarm snapshot URL and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_FindAlarmSnapShotURL](#_获取告警联动抓图URL列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindAlarmSnapShotURL](#_获取告警联动抓图URL列表) and [NETDEV\_FindNextAlarmSnapShotURL](#_逐个查找获取到的URL信息) interfaces.
* Call the [NETDEV\_FindCloseAlarmSnapShotURL](#_关闭告警联动抓图URL列表查找) interface to release resource and stop the search after alarm snapshot URL is obtained.

**See also:**

[NETDEV\_FindAlarmSnapShotURL](#_获取告警联动抓图URL列表), [NETDEV\_FindNextAlarmSnapShotURL](#_逐个查找获取到的URL信息)

### Save alarm snapshot

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SaveSnapShotFile  (  LPVOID lpUserID,  [LPNETDEV\_PIC\_FILE\_INFO\_S](#_抓拍图片信息结构体) pstPicFileInfo  ); |

**Interface description:**

Save alarm snapshot

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstPicFileInfo | IN | Alarm image info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_FindAlarmSnapShotURL](#_获取告警联动抓图URL列表), [NETDEV\_FindNextAlarmSnapShotURL](#_逐个查找获取到的URL信息), [NETDEV\_FindCloseAlarmSnapShotURL](#_关闭告警联动抓图URL列表查找)

## Alarm Message

### Register the callback function to receive alarm messages

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetAlarmCallBack\_V30  (  LPVOID lpUserID,  NETDEV\_AlarmMessCallBack\_PF\_V30 cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive alarm messages

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| cbAlarmMessCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_AlarmMessCallBack\_PF\_V30)  (  LPVOID lpUserID,  [LPNETDEV\_REPORT\_INFO\_S](#_告警/事件上报信息) pstReportInfo,  LPVOID lpBuf,  INT32 dwBufLen,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| pstReportInfo | IN | Reported info |
| lpBuf | IN | Buffer pointer |
| dwBufLen | IN | Length of alarm info struct |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure.

Remarks:

* To close the callback function, set the second parameter to NULL.

### Register the callback function to receive alarm messages

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetAlarmCallBack\_V40  (  LPVOID lpUserID,  NETDEV\_AlarmCallBack\_PF\_V40 cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive alarm messages

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| cbAlarmMessCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \* NETDEV\_AlarmCallBack\_PF\_V40)  (  LPVOID lpUserID,  [LPNETDEV\_REPORT\_INFO\_S](#_告警/事件上报信息) pstReportInfo,  CHAR\* szReference,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| pstReportInfo | IN | Reported info |
| szReference | IN | Descriptive information |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure.

Remarks:

To close the callback function, set the second parameter to NULL.

### Get device alarm info proactively

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_PullAlarm  (  LPVOID lpUserID,  INT32 dwPullWaitTime,  INT32 \*pdwListCnt,  [LPNETDEV\_PULLALARM\_INFO\_S](#_拉告警信息) pstPullAlarmList  ); |

**Interface description:**

Get device alarm info proactively

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwPullWaitTime | IN | Alarm wait time (unit: s) |
| pdwListCnt | INOUT | Max alarm number. An input parameter means the alarm list size, and an output parameter means the actual number of alarms. |
| pstPullAlarmList | OUT | Alarm list. Memory needs to be allocated in advance |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

## People Flow Counting

### Get people counting list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindTrafficStatisticInfoList  (  LPVOID lpUserID,  UINT32 udwSearchID  ); |

**Interface description:**

Get people counting list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| udwSearchID | IN | People counting search ID |

**Return value:**

Search handle ([NETDEV\_FindNextTrafficStatisticInfo](#_逐个查询获取到的客流量信息), [NETDEV\_FindCloseTrafficStatisticInfo](#_关闭查找，释放资源)). NULL means failure.

**See also:**

[NETDEV\_FindNextTrafficStatisticInfo](#_逐个查询获取到的客流量信息), [NETDEV\_FindCloseTrafficStatisticInfo](#_关闭查找，释放资源)

### Get people counting info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextTrafficStatisticInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_TRAFFIC\_STATISTICS\_INFO\_S](#_客流量统计) pstStatisticInfo  ); |

**Interface description:**

Get people counting info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| pstStatisticInfo | OUT | People counting statistics |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

[NETDEV\_FindTrafficStatisticInfoList](#_获取客流量统计信息列表),  [NETDEV\_FindCloseTrafficStatisticInfo](#_关闭查找，释放资源)

### Stop the search and release resource

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseTrafficStatisticInfo(IN LPVOID lpFindHandle) |

**Interface description:**

Stop the search and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

[NETDEV\_FindTrafficStatisticInfoList](#_获取客流量统计信息列表), [NETDEV\_FindNextTrafficStatisticInfo](#_逐个查询获取到的客流量信息)

### Get people counting statistics

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetTrafficStatistic  (  LPVOID lpUserID,  [LPNETDEV\_TRAFFIC\_STATISTICS\_COND\_S](#_客流量统计命令) pstStatisticCond,  [LPNETDEV\_TRAFFIC\_STATISTICS\_DATA\_S](#_客流量统计_1) pstTrafficStatistic  ); |

**Interface description:**

Get people counting statistics

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstStatisticCond | IN | People counting command |
| pstTrafficStatistic | OUT | People counting |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

### Reset people counting statistics

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ResetPassengerFlow  (  LPVOID lpUserID,  INT32 dwChannelID  )； |

**Interface description:**

Reset people counting statistics

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

### Get time interval of reporting people counting statistics

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetTrafficStatisticReportTime  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_TRAFFIC\_STATISTICS\_REPORT\_TIME\_S](#_客流量上报时间间隔) pstReportTimeValue  ); |

**Interface description:**

Get time interval of reporting people counting statistics

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwChannelID | IN | Channel ID |
| pstReportTimeValue | OUT | Time interval of reporting people counting statistics |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

### Start getting people counting statistics

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StartTrafficStatistic  (  LPVOID lpUserID,  [LPNETDEV\_TRAFFIC\_STATISTICS\_COND\_S](#_客流量统计命令) pstStatisticCond,  UINT32\* pudwSearchID  ); |

**Interface description:**

Start getting people counting statistics

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| pstStatisticCond | IN | People counting statistics structure |
| pudwSearchID | OUT | People counting search ID |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

[NETDEV\_StopTrafficStatistic](#_停止客流量查询)

### Stop getting people counting statistics

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopTrafficStatistic  (  LPVOID lpUserID,  UINT32 udwSearchID  ); |

**Interface description:**

**Stop getting people counting statistics**

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| udwSearchID | IN | People counting search ID |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

[NETDEV\_StartTrafficStatistic](#_开始客流量统计查询)

### Start getting multi-channel people counting statistics

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StartMultiTrafficStatistic  (  LPVOID lpUserID,  [LPNETDEV\_MULTI\_TRAFFIC\_STATISTICS\_COND\_S](#_多通道客流量统计命令) pstStatisticCond,  UINT32\* pudwSearchID  ); |

**Interface description:**

Start getting multi-channel people counting statistics

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| pstStatisticCond | IN | People counting statistics structure |
| pudwSearchID | OUT | People counting search ID |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

### Get people counting progress

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetTrafficStatisticProgress  (  LPVOID lpUserID,  UINT32 udwSearchID,  UINT32\* pudwProgress  ); |

**Interface description:**

Get people counting progress

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login ID |
| udwSearchID | IN | People counting search ID |
| pudwProgress | OUT | People counting progress |

**Return value:**

TRUE means success, other values mean failure.

**See also:**

### Register the callback function to receive people counting statistics

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPassengerFlowStatisticCallBack  (  LPVOID lpUserID,  NETDEV\_PassengerFlowStatisticCallBack\_PF cbPassengerFlowStatisticCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive people counting statistics

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| cbPassengerFlowStatisticCallBack | IN | Callback function to register |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL \*NETDEV\_PassengerFlowStatisticCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_PASSENGER\_FLOW\_STATISTIC\_DATA\_S](#_客流量统计信息) pstPassengerFlowData,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstPassengerFlowData | IN | People counting statistics |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure.

Remarks:

* To close the callback function, set the second parameter to NULL.

**See also:**

## Monitoring Task

### Add person

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddPersonInfo  (  LPVOID lpUserID,  UINT32 udwPersonLibID,  [LPNETDEV\_PERSON\_INFO\_LIST\_S](#_人员信息列表结构体) pstPersonInfoList,  [LPNETDEV\_PERSON\_RESULT\_LIST\_S](#_人员信息结果列表结构体) pstPersonResultList  ); |

**Interface description:**

Add person

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwPersonLibID | IN | Person library ID |
| pstPersonInfoList | IN | Person list |
| pstPersonResultList | OUT | Person list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_DeletePersonInfo](#_删除指定的人员信息), [NETDEV\_ModifyPersonInfo](#_修改指定的人员信息), [NETDEV\_DeletePersonInfoList](#_批量删除人员信息)

### Delete person

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeletePersonInfo  (  LPVOID lpUserID,  UINT32 udwPersonLibID,  UINT32 udwPersonID,  UINT32 udwLastChange  ); |

**Interface description:**

Delete person

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwPersonLibID | IN | Person library ID |
| udwPersonID | IN | Person ID |
| udwLastChange | IN | Last time modified |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_ AddPersonInfo](#_新增指定的人员信息), [NETDEV\_ModifyPersonInfo](#_修改指定的人员信息), [NETDEV\_DeletePersonInfoList](#_批量删除人员信息)

### Add a face monitoring task

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddPersonMonitorInfo  (  LPVOID lpUserID,  [LPNETDEV\_MONITION\_INFO\_S](#_布控任务信息结构体) pstMonitorInfo,  [LPNETDEV\_MONITOR\_RESULT\_INFO\_S](#_添加布控返回的布控信息列表结构体) pstMonitorResultInfo  ); |

**Interface description:**

Add a face monitoring task

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstMonitorInfo | INOUT | Monitoring task info. The monitoring task ID is returned if the request is successful. |
| pstMonitorResultInfo | INOUT | Monitoring info returned for the added monitoring task. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error. pstMonitorResultInfo->udwChannelNum must not be less than pstMonitorInfo stMonitorRuleInfo.udwChannelNum.

**See also:**

[NETDEV\_DeletePersonMonitorInfo](#_删除单个人脸布控任务)

### Delete a face monitoring task

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ DeletePersonMonitorInfo  (  LPVOID lpUserID,  UINT32 udwID  ); |

**Interface description:**

Delete a face monitoring task

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwID | IN | Monitoring ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_AddPersonMonitorInfo](#_新增单个人脸布控任务)

### Delete face monitoring tasks in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ NETDEV\_BatchDeletePersonMonitorInfo  (  LPVOID lpUserID,  [LPNETDEV\_BATCH\_OPERATOR\_LIST\_S](#_删除人脸布控返回的信息列表结构体) pstResultList  ); |

**Interface description:**

Delete face monitoring tasks in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstResultList | INOUT | Returned list. Needs to input all the monitoring task IDs to be deleted. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

### Create person library info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_CreatePersonLibInfo  (  LPVOID lpUserID,  [LPNETDEV\_LIB\_INFO\_S](#_人员库信息结构体) pstPersonLibInfo,  UINT32 \*pudwID  ); |

**Interface description:**

Create person library info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstPersonLibInfo | IN | Person library info |
| pudwID | OUT | ID of the created library. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_ ModifyPersonLibInfo](#_修改人员库信息), [NETDEV\_DeletePersonLibInfo](#_删除指定的人员库)

### Modify person library info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyPersonLibInfo  (  LPVOID lpUserID,  [LPNETDEV\_PERSON\_LIB\_LIST\_S](#_人员库信息列表结构体) pstPersonLibList  ); |

**Interface description:**

Modify person library info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstPersonLibList | IN | Person library list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_CreatePersonLibInfo](#_创建人员库信息), [NETDEV\_DeletePersonLibInfo](#_删除指定的人员库)

### Delete a person library

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeletePersonLibInfo  (  LPVOID lpUserID,  UINT32 udwPersonLibID,  [LPNETDEV\_DELETE\_DB\_FLAG\_INFO\_S](#_删除库标志位结构体) pstFlagInfo  ); |

**Interface description:**

Delete a person library

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwPersonLibID | IN | Person library ID |
| pstFlagInfo | IN | Library deleting flag |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_ CreatePersonLibInfo](#_创建人员库信息), [NETDEV\_ ModifyPersonLibInfo](#_修改人员库信息)

### Get configuration of a face monitoring task

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetPersonMonitorRuleInfo  (  LPVOID lpUserID,  [LPNETDEV\_MONITION\_INFO\_S](#_布控任务信息结构体) pstMonitorInfo  ); |

**Interface description:**

Get configuration of a face monitoring task

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstMonitorInfo | INOUT | Monitoring task info. Input monitoring task ID, and output monitoring task info if successful. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error. The pudwMonitorChlIDList in Monitoring Task Info > Face Monitoring Task Configuration is requested by the upper layer. When the interface calling failed and the error code is NETDEV\_E\_NEED\_MORE\_MEMORY, use udwChannelNum to decide whether the requested memory is insufficient.

**See also:**

[NETDEV\_ SetPersonMonitorRuleInfo](#_设置单个人脸布控任务配置信息)

### Set a face monitoring task

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ SetPersonMonitorRuleInfo  (  LPVOID lpUserID,  [LPNETDEV\_MONITION\_INFO\_S](#_布控任务信息结构体) pstMonitorInfo  ); |

**Interface description:**

Set a face monitoring task

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstMonitorInfo | IN | Monitoring task info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_ GetPersonMonitorRuleInfo](#_查询单个人脸布控任务配置信息)

### Get capacity info of all person libraries

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetPersonLibCapacity  (  LPVOID lpUserID,  INT32 dwTimeOut,  [LPNETDEV\_PERSON\_LIB\_CAP\_LIST\_S](#_所有人员库的容量信息结构体) pstCapacityList  ); |

**Interface description:**

Get capacity info of all person libraries

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwTimeOut | IN | Connection timeout value. |
| pstCapacityList | OUT | Capacity of all person libraries |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

### Search person info with criteria

#### Get person list

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindPersonInfoList  (  LPVOID lpUserID,  UINT32 udwPersonLibID,  [LPNETDEV\_PERSON\_QUERY\_INFO\_S](#_人员信息查询条件结构体) pstQueryInfo,  [LPNETDEV\_BATCH\_OPERATE\_BASIC\_S](#_批量查询返回的基本信息结构体) pstQueryResultInfo  ); |

**Interface description:**

Search person list with criteria

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwPersonLibID | IN | Person library ID |
| pstQueryInfo | IN | Face info search criteria |
| pstQueryResultInfo | OUT | Returned person info |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextPersonInfo](#_逐个获取查找到的人员信息) and [NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextPersonInfo](#_逐个获取查找到的人员信息) and [NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextPersonInfo](#_逐个获取查找到的人员信息) interface repeatedly to get the next person info.
* Call the [NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找) interface to release resource and stop the search after person info is obtained.

**See also:**

[NETDEV\_FindNextPersonInfo](#_逐个获取查找到的人员信息), [NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找)

#### Get person info one by one

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextPersonInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_PERSON\_INFO\_S](#_人员信息结构体) pstPersonInfo  ); |

**Interface description:**

Get person info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | File search handle |
| pstPersonInfo | OUT | Person info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindPersonInfoList](#_查询人员信息列表) and [NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找) interface to release resource and stop the search after person info is obtained.

**See also:**

[NETDEV\_FindPersonInfoList](#_查询人员信息列表), [NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找)

#### Stop searching person info

|  |
| --- |
| BOOL STDCALL NETDEV\_FindClosePersonInfoList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching person info and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | File search handle, which is the return value from the [NETDEV\_FindPersonInfoList](#_查询人员信息列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindPersonInfoList](#_查询人员信息列表) and [NETDEV\_FindNextPersonInfo](#_逐个获取查找到的人员信息) interfaces.
* Call the [NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找) interface to release resource and stop the search after person info is obtained.

**See also:**

[NETDEV\_FindPersonInfoList](#_查询人员信息列表), [NETDEV\_FindNextPersonInfo](#_逐个获取查找到的人员信息)

### Get all the existing person libraries

#### Search all the person libraries

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindPersonLibList  (  LPVOID lpUserID  ); |

**Interface description:**

Search all the person libraries

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |

**Return value:**

Null means failure, other values are used as the input parameter of [NETDEV\_FindNextPersonLibInfo](#_逐个获取查找到的人脸库信息) and [NETDEV\_FindClosePersonLibList](#_关闭人脸库信息查找). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextPersonLibInfo](#_逐个获取查找到的人脸库信息) and [NETDEV\_FindClosePersonLibList](#_关闭人脸库信息查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextPersonLibInfo](#_逐个获取查找到的人脸库信息) interface repeatedly to get the next library info.
* Call the [NETDEV\_FindClosePersonLibList](#_关闭人脸库信息查找) interface to release resource and stop the search after face library info is obtained.

**See also:**

[NETDEV\_FindNextPersonLibInfo](#_逐个获取查找到的人脸库信息), [NETDEV\_FindClosePersonLibList](#_关闭人脸库信息查找)

#### Get person library info one by one

|  |
| --- |
| BOOL STDCALL NETDEV\_ FindNextPersonLibInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_LIB\_INFO\_S](#_人员库信息结构体) pstPersonLibInfo  ); |

**Interface description:**

Get person library info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| pstPersonLibInfo | OUT | Face library info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindPersonLibList](#_查询所有已创建的人员库信息) and [NETDEV\_FindClosePersonLibList](#_关闭人脸库信息查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindClosePersonLibList](#_关闭人脸库信息查找) interface to release resource and stop the search after face library info is obtained.

**See also:**

[NETDEV\_FindPersonLibList](#_查询所有已创建的人员库信息), [NETDEV\_FindClosePersonLibList](#_关闭人脸库信息查找)

#### Stop searching face library info

|  |
| --- |
| BOOL STDCALL NETDEV\_FindClosePersonLibList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching person info and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindPersonInfoList](#_查询所有已创建的人员库信息) and [NETDEV\_FindNextPersonInfo](#_逐个获取查找到的人脸库信息) interfaces.
* Call the [NETDEV\_FindClosePersonInfoList](#_关闭人脸库信息查找) interface to release resource and stop the search after face library info is obtained.

**See also:**

[NETDEV\_FindPersonLibList](#_查询所有已创建的人员库信息), [NETDEV\_FindNextPersonInfo](#_逐个获取查找到的人脸库信息)

### Search all the face monitoring tasks

#### Search all the face monitoring tasks

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindPersonMonitorList  (  LPVOID lpUserID,  UINT32 udwChannelID,  [LPNETDEV\_MONITOR\_QUERY\_INFO\_S](#_布控信息查询条件结构体) pstQueryInfo  ); |

**Interface description:**

Get the face monitoring task list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwChannelID | IN | Channel ID. Used only when getting monitoring info of NVR channels. |
| pstQueryInfo | IN | Search criteria. Supported by NVR only. |

**Return value:**

Null means failure, other values are used as the input parameter of [NETDEV\_FindNextPersonMonitorInfo](#_逐个获取查找到的布控任务信息) and [NETDEV\_FindClosePersonMonitorList](#_关闭布控任务信息查找). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextPersonMonitorInfo](#_逐个获取查找到的布控任务信息) and [NETDEV\_FindClosePersonMonitorList](#_关闭布控任务信息查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextPersonMonitorInfo](#_逐个获取查找到的布控任务信息) interface repeatedly to get the next task info.
* Call the [NETDEV\_FindClosePersonMonitorList](#_关闭布控任务信息查找) interface to release resource and stop the search after monitoring task info is obtained.

**See also:**

[NETDEV\_FindNextPersonMonitorInfo](#_逐个获取查找到的布控任务信息), [NETDEV\_FindClosePersonMonitorList](#_关闭布控任务信息查找)

#### Get monitoring task info one by one

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextPersonMonitorInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_MONITION\_INFO\_S](#_布控任务信息结构体) pstMonitorInfo  ); |

**Interface description:**

Get monitoring task info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| pstMonitorInfo | OUT | Monitoring task info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindPersonMonitorList](#_查询所有人脸布控任务列表) and [NETDEV\_FindClosePersonMonitorList](#_关闭布控任务信息查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindClosePersonMonitorList](#_关闭布控任务信息查找) interface to release resource and stop the search after monitoring task info is obtained.

**See also:**

[NETDEV\_FindPersonMonitorList](#_查询所有人脸布控任务列表), [NETDEV\_FindClosePersonMonitorList](#_关闭布控任务信息查找)

#### Stop searching monitoring task info

|  |
| --- |
| BOOL STDCALL NETDEV\_FindClosePersonMonitorList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching monitoring task info and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindPersonMonitorList](#_查询所有人脸布控任务列表) and [NETDEV\_FindNextPersonMonitorInfo](#_逐个获取查找到的布控任务信息) interfaces.
* Call the [NETDEV\_FindClosePersonMonitorList](#_关闭布控任务信息查找) interface to release resource and stop the search after monitoring task info is obtained.

**See also:**

[NETDEV\_FindPersonMonitorList](#_查询所有人脸布控任务列表), [NETDEV\_FindNextPersonMonitorInfo](#_逐个获取查找到的布控任务信息)

### Add a vehicle library

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddVehicleLibInfo  (  LPVOID lpUserID,  [LPNETDEV\_LIB\_INFO\_S](#_人员库信息结构体) pstVehicleLibInfo  ); |

**Interface description:**

Add a vehicle library

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstVehicleLibInfo | INOUT | Vehicle library info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_ModifyVehicleLibInfo](#_修改指定的车辆库信息), [NETDEV\_DeleteVehicleLibInfo](#_删除指定的车辆库信息)

### Delete a vehicle library

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteVehicleLibInfo  (  LPVOID lpUserID,  UINT32 udwVehicleLibID,  [LPNETDEV\_DELETE\_DB\_FLAG\_INFO\_S](#_删除库标志位结构体) pstDelLibFlag  ); |

**Interface description:**

Delete a vehicle library

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwVehicleLibID | IN | Vehicle library ID |
| pstDelLibFlag | IN | Library deleting flag |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_AddVehicleLibInfo](#_新增单个车辆库信息), [NETDEV\_ModifyVehicleLibInfo](#_修改指定的车辆库信息)

### Modify a vehicle library

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyVehicleLibInfo  (  LPVOID lpUserID,  [LPNETDEV\_PERSON\_LIB\_LIST\_S](#_人员库信息列表结构体) pstVehicleLibList  ); |

**Interface description:**

Modify a vehicle library

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstVehicleLibList | IN | Vehicle library list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_AddVehicleLibInfo](#_新增单个车辆库信息), [NETDEV\_DeleteVehicleLibInfo](#_删除指定的车辆库信息)

### Add vehicle members in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddVehicleMemberList  (  LPVOID lpUserID,  UINT32 udwLibID,  [LPNETDEV\_VEHICLE\_INFO\_LIST\_S](#_车辆信息列表结构体) pstVehicleMemberList,  [LPNETDEV\_BATCH\_OPERATOR\_LIST\_S](#_删除人脸布控返回的信息列表结构体) pstResultList  ); |

**Interface description:**

Add vehicle members in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwLibID | IN | Vehicle library ID |
| pstVehicleMemberList | IN | Vehicle list |
| pstResultList | OUT | Returned batch operation results |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_DelVehicleMemberList](#_批量删除车辆成员信息)

### Delete vehicle members in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DelVehicleMemberList  (  LPVOID lpUserID,  UINT32 udwLib,  [LPNETDEV\_VEHICLE\_INFO\_LIST\_S](#_车辆信息列表结构体) pstVehicleMemberList,  [LPNETDEV\_BATCH\_OPERATOR\_LIST\_S](#_删除人脸布控返回的信息列表结构体) pstBatchList  ); |

**Interface description:**

Delete vehicle members in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwLib | IN | Library ID |
| pstVehicleMemberList | IN | Vehicle member list |
| pstBatchList | OUT | Returned batch operation results |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_AddVehicleMemberList](#_批量添加车辆成员信息)

### Delete a vehicle member

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DelVehicleInfo  (  LPVOID lpUserID,  UINT32 udwLibID,  UINT32 udwVehicleID  ); |

**Interface description:**

Delete a vehicle member

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwLibID | IN | Vehicle library ID |
| udwVehicleID | IN | Vehicle ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

### Add a vehicle monitoring task

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddVehicleMonitorInfo  (  LPVOID lpUserID,  [LPNETDEV\_MONITION\_INFO\_S](#_布控任务信息结构体) pstMonitorInfo  ); |

**Interface description:**

Add a vehicle monitoring task

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstMonitorInfo | INOUT | Monitoring task info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_DeleteVehicleMonitorInfo](#_批量删除车辆布控任务)

### Delete vehicle monitoring tasks in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteVehicleMonitorInfo  (  LPVOID lpUserID,  [LPNETDEV\_BATCH\_OPERATOR\_LIST\_S](#_删除人脸布控返回的信息列表结构体) pstBatchList  ); |

**Interface description:**

Delete vehicle monitoring tasks in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstBatchList | INOUT | Vehicle monitoring task list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_AddVehicleMonitorInfo](#_新增单个车辆布控任务)

### Get detailed info about a vehicle member

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetVehicleMemberInfo  (  LPVOID lpUserID,  UINT32 udwVehicleID,  [LPNETDEV\_VEHICLE\_DETAIL\_INFO\_S](#_车辆成员信息) pstVehicleDetailInfo  ); |

**Interface description:**

Get detailed info about a vehicle member

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwVehicleID | IN | Vehicle ID |
| pstVehicleDetailInfo | OUT | Detailed vehicle info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

### Modify vehicle info in a vehicle library

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyVehicleMemberInfo  (  LPVOID lpUserID,  UINT32 udwVehicleLibID,  [LPNETDEV\_VEHICLE\_INFO\_LIST\_S](#_车辆信息列表结构体) pstVehicleMemberList,  [LPNETDEV\_BATCH\_OPERATOR\_LIST\_S](#_删除人脸布控返回的信息列表结构体) pstResultList  ); |

**Interface description:**

Modify vehicle info in a vehicle library

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwVehicleLibID | IN | Vehicle library ID |
| pstVehicleMemberList | IN | Detailed vehicle info |
| pstResultList | OUT | Returned batch operation results |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

### Get configuration of a vehicle monitoring task

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetVehicleMonitorInfo  (  LPVOID lpUserID,  UINT32 udwID,  [LPNETDEV\_MONITION\_RULE\_INFO\_S](#_布控任务配置信息结构体) pstMonitorInfo  ); |

**Interface description:**

Get configuration of a vehicle monitoring task

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwID | IN | Vehicle monitoring ID |
| pstMonitorInfo | OUT | Monitoring task info. Input monitoring task ID, and output monitoring task info if successful. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* The memory for vehicle comparison images is assigned by user. The size of image receiving buffer needs to be specified as the input parameter. The actual size needed will be returned if calling the interface failed.
* Error code: NETDEV\_E\_NEED\_MORE\_MEMORY User-allocated memory is insufficient
* If there is no vehicle comparison image, the size of vehicle comparison image will be set to 0.

**See also:**

[NETDEV\_SetVehicleMonitorInfo](#_设置单个车辆布控任务配置信息)

### Set a vehicle monitoring task

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetVehicleMonitorInfo  (  LPVOID lpUserID,  UINT32 udwID,  [LPNETDEV\_MONITION\_RULE\_INFO\_S](#_布控任务配置信息结构体) pstMonitorInfo  ); |

**Interface description:**

Set a vehicle monitoring task

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwID | IN | Vehicle monitoring ID |
| pstMonitorInfo | IN | Monitoring task info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetVehicleMonitorInfo](#_查询单个车辆布控任务配置信息)

### Assign vehicle members in batches to a vehicle library

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddVehicleLibMember  (  LPVOID lpUserID,  UINT32 udwVehicleLibID,  [LPNETDEV\_BATCH\_OPERATE\_MEMBER\_LIST\_S](#_批量操作成员列表结构体) pstMemberList,  [LPNETDEV\_BATCH\_OPERATOR\_LIST\_S](#_删除人脸布控返回的信息列表结构体) pstBatchResultList  ); |

**Interface description:**

Assign vehicle members in batches to a vehicle library

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwVehicleLibID | IN | Vehicle library ID |
| pstMemberList | IN | Assign vehicle member IDs in batches |
| pstBatchResultList | OUT | Returned batch operation results |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_DeleteVehicleLibMember](#_批量取消指定的车辆库中车辆成员划归)

### Remove vehicle members in batches from a vehicle library

**Interface name:**

|  |
| --- |
| BOOL STDCALLNETDEV\_DeleteVehicleLibMember  (  LPVOID lpUserID,  UINT32 udwVehicleLibID,  [LPNETDEV\_BATCH\_OPERATE\_MEMBER\_LIST\_S](#_批量操作成员列表结构体) pstMemberList,  [LPNETDEV\_BATCH\_OPERATOR\_LIST\_S](#_删除人脸布控返回的信息列表结构体) pstBatchResultList  ); |

**Interface description:**

Remove vehicle members in batches from a vehicle library

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwVehicleLibID | IN | Vehicle library ID |
| pstMemberList | IN | Remove vehicle member IDs in batches |
| pstBatchResultList | OUT | Returned batch operation results |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_AddVehicleLibMember](#_向指定的车辆库中批量划归车辆成员)

### Get vehicle image info of a vehicle recognition record

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetVehicleRecordImageInfo  (  LPVOID lpUserID,  UINT32 udwRecordID,  [LPNETDEV\_FILE\_INFO\_S](#_文件信息结构体) pstFileInfo  ); |

**Interface description:**

Get vehicle image info of a vehicle recognition record

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwRecordID | IN | Vehicle recognition record ID |
| pstFileInfo | INOUT | Vehicle image info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

### Get vehicle library list

#### Get vehicle library list

|  |
| --- |
| LPVOID STDCALL NETDEV\_ FindVehicleLibList  (  LPVOID lpUserID  ); |

**Interface description:**

Get vehicle library list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |

**Return value:**

Null means failure, other values are used as the input parameter of [NETDEV\_FindNextVehicleLibInfo](#_逐个获取查找到的车辆库信息) and [NETDEV\_FindCloseVehicleLibList](#_关闭车辆库信息查找). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextVehicleLibInfo](#_逐个获取查找到的车辆库信息) and [NETDEV\_FindCloseVehicleLibList](#_关闭车辆库信息查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextVehicleLibInfo](#_逐个获取查找到的车辆库信息) interface repeatedly to get the next library info.
* Call the [NETDEV\_FindCloseVehicleLibList](#_关闭车辆库信息查找) interface to release resource and stop the search after vehicle library info is obtained.

**See also:**

[NETDEV\_FindNextVehicleLibInfo](#_逐个获取查找到的车辆库信息), [NETDEV\_FindCloseVehicleLibList](#_关闭车辆库信息查找)

#### Get vehicle library info one by one

|  |
| --- |
| BOOL STDCALL NETDEV\_ FindNextVehicleLibInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_LIB\_INFO\_S](#_人员库信息结构体) pstVehicleLibInfo  ); |

**Interface description:**

Get vehicle library info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| pstVehicleLibInfo | OUT | Vehicle library info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindVehicleLibList](#_查询全部车辆库信息列表) and [NETDEV\_FindCloseVehicleLibList](#_关闭车辆库信息查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseVehicleLibList](#_关闭车辆库信息查找) interface to release resource and stop the search after vehicle library info is obtained.

**See also:**

[NETDEV\_FindVehicleLibList](#_查询全部车辆库信息列表), [NETDEV\_FindCloseVehicleLibList](#_关闭车辆库信息查找)

#### Stop searching vehicle library info

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseVehicleLibList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching vehicle library info and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindVehicleLibList](#_查询全部车辆库信息列表) and [NETDEV\_FindNextVehicleLibInfo](#_逐个获取查找到的车辆库信息) interfaces.
* Call the [NETDEV\_FindCloseVehicleLibList](#_关闭车辆库信息查找) interface to release resource and stop the search after vehicle library info is obtained.

**See also:**

[NETDEV\_FindVehicleLibList](#_查询全部车辆库信息列表), [NETDEV\_FindNextVehicleLibInfo](#_逐个获取查找到的车辆库信息)

### Search vehicle recognition records with criteria

#### Search vehicle recognition records with criteria

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindVehicleRecordInfoList  (  LPVOID lpUserID,  [LPNETDEV\_ALARM\_LOG\_COND\_LIST\_S](#_告警日志查询信息结构体) pstFindCond,  [LPNETDEV\_SMART\_ALARM\_LOG\_RESULT\_INFO\_S](#_告警记录返回信息（人脸识别和车牌识别）结构体) pstResultInfo  ); |

**Interface description:**

Search vehicle recognition records with criteria

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstFindCond | IN | Search criteria |
| pstResultInfo | OUT | Returned records |

**Return value:**

Null means failure, other values are used as the input parameter of [NETDEV\_FindNextVehicleRecordInfo](#_逐个获取查找到的车辆识别记录信息) and [NETDEV\_FindCloseVehicleRecordList](#_关闭车辆识别记录信息查找). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextVehicleRecordInfo](#_逐个获取查找到的车辆识别记录信息) and [NETDEV\_FindCloseVehicleRecordList](#_关闭车辆识别记录信息查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextVehicleRecordInfo](#_逐个获取查找到的车辆识别记录信息) interface repeatedly to get the next record info.
* Call the [NETDEV\_FindCloseVehicleRecordList](#_关闭车辆识别记录信息查找) interface to release resource and stop the search after vehicle recognition record info is obtained.

**See also:**

[NETDEV\_FindNextVehicleRecordInfo](#_逐个获取查找到的车辆识别记录信息), [NETDEV\_FindCloseVehicleRecordList](#_关闭车辆识别记录信息查找)

#### Get vehicle recognition record info one by one

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextVehicleRecordInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_VEHICLE\_RECORD\_INFO\_S](#_车辆识别记录信息结构体) pstRecordInfo  ); |

**Interface description:**

Get vehicle recognition record info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| pstRecordInfo | OUT | Vehicle recognition record info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindVehicleRecordInfoList](#_条件查询车辆识别记录的详细信息) and [NETDEV\_FindCloseVehicleRecordList](#_关闭车辆识别记录信息查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseVehicleRecordList](#_关闭车辆识别记录信息查找) interface to release resource and stop the search after vehicle recognition record info is obtained.

**See also:**

[NETDEV\_FindVehicleRecordInfoList](#_条件查询车辆识别记录的详细信息), [NETDEV\_FindCloseVehicleRecordList](#_关闭车辆识别记录信息查找)

#### Stop searching vehicle recognition record info

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseVehicleRecordList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching vehicle recognition record info and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindVehicleRecordInfoList](#_条件查询车辆识别记录的详细信息) and [NETDEV\_FindNextVehicleRecordInfo](#_逐个获取查找到的车辆识别记录信息) interfaces.
* Call the [NETDEV\_FindCloseVehicleRecordList](#_关闭车辆识别记录信息查找) interface to release resource and stop the search after vehicle recognition record info is obtained.

**See also:**

[NETDEV\_FindVehicleRecordInfoList](#_条件查询车辆识别记录的详细信息), [NETDEV\_FindNextVehicleRecordInfo](#_逐个获取查找到的车辆识别记录信息)

### Get all vehicle monitoring tasks

#### Get all vehicle monitoring tasks

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindVehicleMonitorList  (  LPVOID lpUserID  ); |

**Interface description:**

Get all vehicle monitoring tasks

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |

**Return value:**

Null means failure, other values are used as the input parameter of [NETDEV\_FindNextVehicleMonitorInfo](#_逐个获取查找到的车辆布控任务信息) and [NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextVehicleMonitorInfo](#_逐个获取查找到的车辆布控任务信息) and [NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextVehicleMonitorInfo](#_逐个获取查找到的车辆布控任务信息) interface repeatedly to get the next task info.
* Call the [NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找) interface to release resource and stop the search after vehicle monitoring task info is obtained.

**See also:**

[NETDEV\_FindNextVehicleMonitorInfo](#_逐个获取查找到的车辆布控任务信息), [NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找)

#### Get vehicle monitoring task info one by one

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextVehicleMonitorInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_MONITION\_INFO\_S](#_布控任务信息结构体) pstVehicleMonitorInfo  ); |

**Interface description:**

Get vehicle monitoring task info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| pstVehicleMonitorInfo | OUT | Monitoring task info |

**Return value:**

TRUE means success, other values mean failure. The memory for vehicle comparison images is assigned by user. The size of image receiving buffer needs to be specified when used as the input parameter. The actual size needed will be returned if calling the interface failed. If there is no vehicle comparison image, the size of vehicle comparison image will be set to 0. The obtained image data must be saved separately, otherwise, memory will be released after the [NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找) interface is called. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindVehicleMonitorList](#_查询车辆识别的所有布控任务) and [NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找) interface to release resource and stop the search after vehicle monitoring task info is obtained.

**See also:**

[NETDEV\_FindVehicleMonitorList](#_查询车辆识别的所有布控任务), [NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找)

#### Stop searching vehicle monitoring task info

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseVehicleMonitorList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching vehicle monitoring task info and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindVehicleMonitorList](#_查询车辆识别的所有布控任务) and [NETDEV\_FindNextVehicleMonitorInfo](#_逐个获取查找到的车辆布控任务信息) interfaces.
* Call the [NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找) interface to release resource and stop the search after vehicle monitoring task info is obtained.

**See also:**

[NETDEV\_FindVehicleMonitorList](#_查询车辆识别的所有布控任务), [NETDEV\_FindNextVehicleMonitorInfo](#_逐个获取查找到的车辆布控任务信息)

### Search vehicle members with criteria

#### Search vehicle members with criteria

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindVehicleMemberDetailList  (  LPVOID lpUserID,  UINT32 udwVehicleLibID,  [LPNETDEV\_PERSON\_QUERY\_INFO\_S](#_人员信息查询条件结构体) pstFindCond,  [LPNETDEV\_BATCH\_OPERATE\_BASIC\_S](#_批量查询返回的基本信息结构体) pstDBMemberList  ); |

**Interface description:**

Search vehicle members with criteria

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwVehicleLibID | IN | Library ID |
| pstFindCond | IN | Search criteria |
| pstDBMemberList | OUT | Returned member info |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextVehicleMemberDetail](#_逐个获取查找到的车辆成员信息) and [NETDEV\_FindCloseVehicleMemberDetail](#_关闭车辆成员信息信息查找).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextVehicleMemberDetail](#_逐个获取查找到的车辆成员信息) and [NETDEV\_FindCloseVehicleMemberDetail](#_关闭车辆成员信息信息查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextVehicleMemberDetail](#_逐个获取查找到的车辆成员信息) interface repeatedly to get the next member info.
* Call the [NETDEV\_FindCloseVehicleMemberDetail](#_关闭车辆成员信息信息查找) interface to release resource and stop the search after vehicle member info is obtained.

**See also:**

[NETDEV\_FindNextVehicleMemberDetail](#_逐个获取查找到的车辆成员信息), [NETDEV\_FindCloseVehicleMemberDetail](#_关闭车辆成员信息信息查找)

#### Get vehicle member info one by one

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextVehicleMemberDetail  (  LPVOID lpFindHandle,  [LPNETDEV\_VEHICLE\_DETAIL\_INFO\_S](#_车辆成员信息) pstVehicleMemberInfo  ); |

**Interface description:**

Get vehicle member info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| pstVehicleMemberInfo | OUT | Vehicle member info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindVehicleMemberDetailList](#_条件查询车辆成员详细信息) and [NETDEV\_FindCloseVehicleMemberDetail](#_关闭车辆成员信息信息查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseVehicleMemberDetail](#_关闭车辆成员信息信息查找) interface to release resource and stop the search after vehicle member info is obtained.

**See also:**

[NETDEV\_FindVehicleMemberDetailList](#_条件查询车辆成员详细信息), [NETDEV\_FindCloseVehicleMemberDetail](#_关闭车辆成员信息信息查找)

#### Stop searching vehicle member info

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseVehicleMemberDetail  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching vehicle member info and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindVehicleMemberDetailList](#_条件查询车辆成员详细信息) and [NETDEV\_FindNextVehicleMemberDetail](#_逐个获取查找到的车辆成员信息) interfaces.
* Call the [NETDEV\_FindCloseVehicleMemberDetail](#_关闭车辆成员信息信息查找) interface to release resource and stop the search after vehicle member info is obtained.

**See also:**

[NETDEV\_FindVehicleMemberDetailList](#_条件查询车辆成员详细信息), [NETDEV\_FindNextVehicleMemberDetail](#_逐个获取查找到的车辆成员信息)

### Modify a person

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyPersonInfo  (  LPVOID lpUserID,  UINT32 udwPersonLibID,  [LPNETDEV\_PERSON\_INFO\_LIST\_S](#_人员信息列表结构体) pstPersonInfoList,  [LPNETDEV\_PERSON\_RESULT\_LIST\_S](#_人员信息结果列表结构体) pstPersonResultList  ); |

**Interface description:**

Modify a person

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwPersonLibID | IN | Person library ID |
| pstPersonInfoList | IN | Person list |
| pstPersonResultList | OUT | Returned person list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_DeletePersonInfo](#_删除指定的人员信息), [NETDEV\_AddPersonInfo](#_新增指定的人员信息), [NETDEV\_DeletePersonInfoList](#_批量删除人员信息)

### Delete persons in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeletePersonInfoList  (  LPVOID lpUserID,  UINT32 udwPersonLibID,  [LPNETDEV\_BATCH\_OPERATE\_MEMBER\_LIST\_S](#_批量操作成员列表结构体) pstIDList,  [LPNETDEV\_BATCH\_OPERATOR\_LIST\_S](#_删除人脸布控返回的信息列表结构体) pstResutList  ); |

**Interface description:**

Delete persons in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| udwPersonLibID | IN | Face library ID |
| pstIDList | IN | Face member list |
| pstResutList | OUT | Returned batch operation results |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_DeletePersonInfo](#_删除指定的人员信息), [NETDEV\_AddPersonInfo](#_新增指定的人员信息), [NETDEV\_ModifyPersonInfo](#_修改指定的人员信息)

## Transparent Transmission of Data

### Create transparent channel

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_SerialStart  (  LPVOID lpUserID,  LPVOID lpInBuffer,  INT64 ulInBufferSize,  NETDEV\_SerialDataCallBack\_PF cbSerialDataCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Create transparent channel;

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| lpInBuffer | IN | Input buffer, which points to serial parameters of transparent channel. See [NETDEV\_SERIAL\_START\_S](#_建立透明通道参数结构体) |
| ulInBufferSize | IN | Input buffer size |
| cbSerialDataCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_SerialDataCallBack\_PF)  (  LPVOID lpSerialHandle,  INT32 dwChannelID,  CHAR\* pRecvDataBuffer,  INT64 ulBufSize,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpSerialHandle | IN | Handle returned when creating transparent channel |
| dwChannelID | IN | Channel ID |
| pRecvDataBuffer | IN | Pointer to data storing buffer |
| ulBufSize | IN | Data size |
| lpUserData | IN | Parameter set by user |

**Return value:**

0 means failure, other values mean the returned handle value of transparent channel. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SerialSend](#_通过透明通道向设备串口发送数据), [NETDEV\_SerialStop](#_断开透明通道)

### Send data to device’s serial port through transparent channel

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SerialSend  (  LPVOID lpSerialHandle,  INT32 dwChannelID,  CHAR\* pSendBuf,  INT64 ulBufSize  ); |

**Interface description:**

Send data to device’s serial port through transparent channel

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpSerialHandle | IN | Handle returned when creating transparent channel, which is the return value from the [NETDEV\_SerialStart](#_建立透明通道) interface. |
| dwChannelID | IN | Serial port number |
| pSendBuf | IN | Pointer to data sending buffer |
| ulBufSize | IN | Buffer size |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SerialStart](#_建立透明通道), [NETDEV\_SerialStop](#_断开透明通道)

### Disconnect transparent channel

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SerialStop  (  LPVOID lpSerialHandle  ); |

**Interface description:**

Disconnect transparent channel

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpSerialHandle | IN | Handle returned when creating transparent channel, which is the return value from the [NETDEV\_SerialStart](#_建立透明通道) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SerialStart](#_建立透明通道), [NETDEV\_SerialSend](#_通过透明通道向设备串口发送数据)

## User Configuration

### Add user info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_CreateUser  (  LPVOID lpUserID,  [LPNETDEV\_USER\_DETAIL\_INFO\_S](#_用户信息结构体) pstUserDetailInfo  ); |

**Interface description:**

Add user info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstUserDetailInfo | IN | User info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_ModifyUser](#_修改用户信息), [NETDEV\_DeleteUser](#_删除用户信息)

### Modify user info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyUser  (  LPVOID lpUserID,  [LPNETDEV\_USER\_MODIFY\_DETAIL\_INFO\_S](#_修改用户详细信息结构体) pstUserInfo  ); |

**Interface description:**

Modify user info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstUserInfo | IN | User info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_CreateUser](#_创建用户信息), [NETDEV\_DeleteUser](#_删除用户信息)

### Delete user info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteUser  (  LPVOID lpUserID,  CHAR \*pszUserName  ); |

**Interface description:**

Delete user info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pszUserName | IN | Username |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_CreateUser](#_创建用户信息), [NETDEV\_ModifyUser](#_修改用户信息)

### Change current user’s password

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ChangePassword  (  LPVOID lpUserID,  CHAR\* pszNewPasswd  ); |

**Interface description:**

Change current user’s password

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pszNewPasswd | IN | New password |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

### Change a specified user’s password

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyPassword  (  LPVOID lpUserID,  [LPNETDEV\_MODIFY\_PASSWORD\_INFO\_S](#_用户密码信息结构体) pstModifyPasswordInfo  ); |

**Interface description:**

Change a specified user’s password

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstModifyPasswordInfo | INOUT | User password info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

### Get current device password

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDevPassword  (  LPVOID lpUserID,  CHAR \*pszPassword  ); |

**Interface description:**

Get current password of device

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pszPassword | OUT | Password. Memory length must be greater than or equal to NETDEV\_LEN\_64. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

### Get detailed info of a user

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetUserDetailInfo  (  LPVOID lpUserID,  [LPNETDEV\_USER\_DETAIL\_INFO\_S](#_用户信息结构体) pstUserDetailInfo  ); |

**Interface description:**

Get detailed info of a user

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstUserDetailInfo | INOUT | User info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

### Get detailed info of a user

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetUserDetailList  (  LPVOID lpUserID,  [LPNETDEV\_USER\_DETAIL\_LIST\_S](#_用户信息列表结构体) pstUserDetailList  ); |

**Interface description:**

Get detailed info of a user

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstUserDetailList | OUT | User list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

## Organization Management

### Add Organization

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddOrgInfo  (  LPVOID lpUserID,  [LPNETDEV\_ORG\_INFO\_S](#_组织信息结构体) pstOrgInfo,  INT32 \*dwOrgID  ); |

**Interface description:**

Add Organization

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstOrgInfo | IN | Organization info |
| dwOrgID | OUT | Organization ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_ModifyOrgInfo](#_修改组织), [NETDEV\_BatchDeleteOrgInfo](#_批量删除组织)

### Modify Organization

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyOrgInfo  (  LPVOID lpUserID,  [LPNETDEV\_ORG\_INFO\_S](#_组织信息结构体) pstOrgInfo  ); |

**Interface description:**

Modify Organization

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstOrgInfo | IN | Organization info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_AddOrgInfo](#_添加组织), [NETDEV\_BatchDeleteOrgInfo](#_批量删除组织)

### Delete organizations in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_BatchDeleteOrgInfo  (  LPVOID lpUserID,  [LPNETDEV\_DEL\_ORG\_INFO\_S](#_待删除组织信息结构体) pstOrgDelInfo,  [LPNETDEV\_ORG\_BATCH\_DEL\_INFO\_S](#_删除组织响应信息结构体) pstOrgDelResultInfo  ); |

**Interface description:**

Delete organizations in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstOrgDelInfo | IN | Target organizational info |
| pstOrgDelResultInfo | OUT | Returned batch operation results |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_AddOrgInfo](#_添加组织), [NETDEV\_ModifyOrgInfo](#_修改组织)

### Modify channel by organization ID

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyOrgChnInfo  (  LPVOID lpUserID,  [LPNETDEV\_ORG\_CHN\_SHORT\_INFO\_S](#_组织下通道信息结构体) pstOrgChnShortInfo  ); |

**Interface description:**

Modify channel by organization ID.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstOrgChnShortInfo | OUT | Info about channels in the organization |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetOrgChnInfo](#_根据组织ID获取通道)

### Get channel by organization ID

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetOrgChnInfo  (  LPVOID lpUserID,  INT32 dwOrgID,  [LPNETDEV\_ORG\_CHN\_SHORT\_INFO\_S](#_组织下通道信息结构体) pstOrgChnShortInfo  ); |

**Interface description:**

Get channel by organization ID

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwOrgID | IN | Organization ID |
| pstOrgChnShortInfo | OUT | Info about channels in the organization |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Call this interface twice. First call the interface to get the number of channels and return the NETDEV\_E\_NEED\_MORE\_MEMORY error code; after memory is allocated dynamically according to the number of channels, call this interface again to get channel info.

**See also:**

[NETDEV\_ModifyOrgChnInfo](#_根据组织ID修改通道)

### Get channel list by organization ID

#### Get channel list by organization ID

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindOrgChnList  (  LPVOID lpUserID,  INT32 dwOrgID,  INT32 dwChnType  ); |

**Interface description:**

Get channel list by organization ID

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| dwOrgID | IN | Organization ID |
| dwChnType | IN | Channel type. See #[NETDEV\_CHN\_TYPE\_E](#_通道类型) |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextOrgChn](#_逐个获取查找到的组织通道信息) and [NETDEV\_FindCloseOrgChn](#_关闭组织通道信息查找).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextOrgChn](#_逐个获取查找到的组织通道信息) and [NETDEV\_FindCloseOrgChn](#_关闭组织通道信息查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_ FindNextOrgChn](#_逐个获取查找到的组织通道信息) interface repeatedly to get the next channel info.
* Call the [NETDEV\_FindCloseOrgChn](#_关闭组织通道信息查找) interface to release resource and stop the search after organization channel info is obtained.

**See also:**

[NETDEV\_FindNextOrgChn](#_逐个获取查找到的组织通道信息), [NETDEV\_FindCloseOrgChn](#_关闭组织通道信息查找)

#### Get organization channel info one by one

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextOrgChn  (  LPVOID lpFindHandle,  [LPNETDEV\_ORG\_CHN\_INFO\_S](#_组织通道信息结构体) pstOrgChnInfo  ); |

**Interface description:**

Get organization channel info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| pstOrgChnInfo | OUT | Info about channels in the organization |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindOrgChnList](#_通过组织ID查询通道信息列表) and [NETDEV\_FindCloseOrgChn](#_关闭组织通道信息查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseOrgChn](#_关闭组织通道信息查找) interface to release resource and stop the search after organization channel info is obtained.

**See also:**

[NETDEV\_FindOrgChnList](#_通过组织ID查询通道信息列表), [NETDEV\_FindCloseOrgChn](#_关闭组织通道信息查找)

#### Stop searching organization channel info

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseOrgChn  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching organization channel info and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindOrgChnList](#_通过组织ID查询通道信息列表) and [NETDEV\_FindNextOrgChn](#_逐个获取查找到的组织通道信息) interfaces.
* Call the [NETDEV\_FindCloseOrgChn](#_关闭组织通道信息查找) interface to release resource and stop the search after organization channel info is obtained.

**See also:**

[NETDEV\_FindOrgChnList](#_通过组织ID查询通道信息列表), [NETDEV\_FindNextOrgChn](#_逐个获取查找到的组织通道信息)

### Get organization list

#### Get organization list

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindOrgInfoList  (  LPVOID lpUserID,  [LPNETDEV\_ORG\_FIND\_COND\_S](#_查找组织信息列表条件结构体) pstFindCond  ); |

**Interface description:**

Get organization list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstFindCond | IN | Search criteria for getting organization list |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextOrgInfo](#_逐个获取查找到的组织信息) and [NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextOrgInfo](#_逐个获取查找到的组织信息) and [NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_ FindNextOrgInfo](#_逐个获取查找到的组织信息) interface repeatedly to get the next organization info.
* Call the [NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找) interface to release resource and stop the search after organization info is obtained.

**See also:**

[NETDEV\_FindNextOrgInfo](#_逐个获取查找到的组织信息), [NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找)

#### Get organization info one by one

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextOrgInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_ORG\_INFO\_S](#_组织信息结构体) pstOrgInfo  ); |

**Interface description:**

Get organization info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |
| pstOrgInfo | OUT | Organization info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindOrgInfoList](#_查询组织信息列表) and [NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找) interface to release resource and stop the search after organization info is obtained.

**See also:**

[NETDEV\_FindOrgInfoList](#_查询组织信息列表), [NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找)

#### Stop searching organization info

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseOrgInfo  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching organization info and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindOrgInfoList](#_查询组织信息列表) and [NETDEV\_FindNextOrgInfo](#_逐个获取查找到的组织信息) interfaces.
* Call the [NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找) interface to release resource and stop the search after organization info is obtained.

**See also:**

[NETDEV\_FindOrgInfoList](#_查询组织信息列表), [NETDEV\_FindNextOrgInfo](#_逐个获取查找到的组织信息)

## Network Configuration

### Get NAT types

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetNATType  (  CHAR\* pszDomain,  INT32 \*pdwNatType  ); |

**Interface description:**

Get NAT types

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| pszDomain | IN | Domain name |
| pdwNatType | OUT | NAT type. See #[NETDEV\_NAT\_TYPE\_E](#_NAT类型枚举) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

### Get mapped port

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetUpnpNatState  (  LPVOID lpUserID,  [LPNETDEV\_UPNP\_NAT\_STATE\_S](#_网络端口号状态信息结构体) pstNatState  ); |

**Interface description:**

Get mapped port

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstNatState | OUT | Network port status info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SetUpnpNatState](#_设置映射端口)

### Set mapped port

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetUpnpNatState  (  LPVOID lpUserID,  [LPNETDEV\_UPNP\_NAT\_STATE\_S](#_网络端口号状态信息结构体) pstNatState  ); |

**Interface description:**

Set mapped port

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle |
| pstNatState | IN | Network port status info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_GetUpnpNatState](#_获取映射端口)

## Decoding Device

### Get capabilities

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetCapability  (  LPVOID lpUserID,  [LPNETDEV\_XW\_CAP\_INFO\_S](#_XW设备能力集信息结构体) pstCapInfo  ); |

**Interface description:**

Get capabilities.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstCapInfo | OUT | Video wall info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get the number of local encoding channels

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetChannelsNum  (  LPVOID lpUserID,  [LPNETDEV\_XW\_CHANNELS\_NUM\_S](#_通道数量信息结构体) pstChannels  ); |

**Interface description:**

Get the number of local encoding channels

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstChannels | OUT | Channel info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get video channel list

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetChannelsInfoList  (  LPVOID lpUserID,  [LPNETDEV\_XW\_CHANNELS\_LIST\_S](#_解码器通道信息结构体) pstChannelsList  ); |

**Interface description:**

Get video channel list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstChannelsList | INOUT | Encoding channel info. Input the number of channels, and output channel info. Caller allocates memory. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Call the [NETDEV\_XW\_GetChannelsNum](#_获取本地编码通道数量) interface to get the number of channels.

**See also:**

[NETDEV\_XW\_GetChannelsNum](#_获取本地编码通道数量)

### Create video wall configuration

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_CreateTVWallCfg  (  LPVOID lpUserID,  [LPNETDEV\_XW\_TVWALL\_CFG\_S](#_电视墙信息结构体) pstTVWallCfg,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Create video wall configuration

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstTVWallCfg | INOUT | Video wall info. The device returns video wall ID, resource ID OrderNo after creating the video wall. |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Modify video wall configuration

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_ModifyTVWallCfg  (  LPVOID lpUserID,  [LPNETDEV\_XW\_TVWALL\_CFG\_S](#_电视墙信息结构体) pstTVWallCfg,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Modify video wall configuration

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstTVWallCfg | IN | Video wall info. Video wall ID needs to be input in the structure. |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Delete video wall configuration

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteTVWallCfg  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Delete video wall configuration

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get video wall configuration

#### Get video wall configuration list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindTVWallCfgList  (  LPVOID lpUserID  ); |

**Interface description:**

Get video wall configuration list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_XW\_FindNextTVWallCfg](#_逐个获取查找到的电视墙配置信息) and [NETDEV\_XW\_FindCloseTVWallCfg](#_关闭查找电视墙配置信息).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindNextTVWallCfg](#_逐个获取查找到的电视墙配置信息) and [NETDEV\_XW\_FindCloseTVWallCfg](#_关闭查找电视墙配置信息) interfaces.
* After the interface is called successfully, call the [NETDEV\_XW\_FindNextTVWallCfg](#_逐个获取查找到的电视墙配置信息) interface repeatedly to get the next video wall configuration.
* Call the [NETDEV\_XW\_FindCloseTVWallCfg](#_关闭查找电视墙配置信息) interface to release resource after video wall configuration is obtained.

**See also:**

[[NETDEV\_XW\_FindNextTVWallCfg](#_逐个获取查找到的电视墙配置信息)](#_逐个获取查找到的电视墙配置信息), [NETDEV\_XW\_FindCloseTVWallCfg](#_关闭查找电视墙配置信息)

#### Get configuration of video walls one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextTVWallCfg  (  LPVOID lpFindHandle,  [LPNETDEV\_XW\_TVWALL\_CFG\_S](#_电视墙信息结构体) pstTVWallCfg  ); |

**Interface description:**

Get configuration of video walls one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindTVWallCfgList](#_获取电视墙配置列表) interface. |
| pstTVWallCfg | OUT | Video wall configuration |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindTVWallCfgList](#_获取电视墙配置列表) and [NETDEV\_XW\_FindCloseTVWallCfg](#_关闭查找电视墙配置信息) interfaces.
* Call this interface repeatedly to get the next video wall configuration.
* Call the [NETDEV\_XW\_FindCloseTVWallCfg](#_关闭查找电视墙配置信息) interface to release resource after video wall configuration is obtained.

**See also:**

[NETDEV\_XW\_FindTVWallCfgList](#_获取电视墙配置列表), [NETDEV\_XW\_FindCloseTVWallCfg](#_关闭查找电视墙配置信息)

#### Stop searching video wall configuration

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseTVWallCfg  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching video wall configuration and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindTVWallCfgList](#_获取电视墙配置列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindTVWallCfgList](#_获取电视墙配置列表) and [NETDEV\_XW\_FindNextTVWallCfg](#_逐个获取查找到的电视墙配置信息) interfaces.
* Call the [NETDEV\_XW\_FindCloseTVWallCfg](#_关闭查找电视墙配置信息) interface to release resource after video wall configuration is obtained.

**See also:**

[NETDEV\_XW\_FindTVWallCfgList](#_获取电视墙配置列表), [NETDEV\_XW\_FindNextTVWallCfg](#_逐个获取查找到的电视墙配置信息)

### Get configuration of a video wall

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetTVWallCfg  (  LPVOID lpUserID,  UINT32 udwTVWallID,  [LPNETDEV\_XW\_TVWALL\_CFG\_S](#_电视墙信息结构体) pstTVWallCfg  ); |

**Interface description:**

Get configuration of a video wall

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTVWallID | IN | Video wall ID |
| pstTVWallCfg | INOUT | Video wall info. szTVWallCode is required if video wall code exists. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Create a virtual LED

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_CreateVirtualLED  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_VIRTUAL\_LED\_INFO\_S](#_虚拟LED信息结构体) pstVirtualLEDInfo,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Create a virtual LED

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstVirtualLEDInfo | INOUT | Virtual LED info. The device returns LEDID. |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Delete a virtual LED

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteOneVirtualLED  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwLedlID,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Delete a virtual LED

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| udwLedlID | IN | Virtual LED ID |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Configure a virtual LED

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_SetVirtualLED  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_VIRTUAL\_LED\_INFO\_S](#_虚拟LED信息结构体) pstVirtualLEDInfo,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Configure a virtual LED

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstVirtualLEDInfo | IN | Virtual LED info |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get a virtual LED

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetVirtualLED  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_VIRTUAL\_LED\_INFO\_S](#_虚拟LED信息结构体) pstVirtualLEDInfo  ); |

**Interface description:**

Get a virtual LED

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstVirtualLEDInfo | INOUT | Virtual LED info. LED ID is required. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get virtual LED list

#### Get virtual LED list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindVirtualLEDList  (  LPVOID lpUserID,  UINT32 udwTvWallID  ); |

**Interface description:**

Get virtual LED list;

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_XW\_FindNextVirtualLED](#_逐个获取虚拟LED信息) and [NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindNextVirtualLED](#_逐个获取虚拟LED信息) and [NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息) interfaces.
* After the interface is called successfully, call the [NETDEV\_XW\_FindNextVirtualLED](#_逐个获取虚拟LED信息) interface repeatedly to get the next virtual LED info.
* Call the [NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息) interface to release resource after virtual LED info is obtained.

**See also:**

[NETDEV\_XW\_FindNextVirtualLED](#_逐个获取虚拟LED信息), [NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息)

#### Get virtual LED info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextVirtualLED  (  LPVOID lpFindHandle,  [LPNETDEV\_XW\_VIRTUAL\_LED\_INFO\_S](#_虚拟LED信息结构体) pstVirtualLEDInfo  ); |

**Interface description:**

Get virtual LED info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindVirtualLEDList](#_获取虚拟LED列表) interface. |
| pstVirtualLEDInfo | OUT | Virtual LED info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindVirtualLEDList](#_获取虚拟LED列表) and [NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息) interface to release resource after virtual LED info is obtained.

**See also:**

[NETDEV\_XW\_FindVirtualLEDList](#_获取虚拟LED列表), [NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息)

#### Stop searching virtual LED info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseVirtualLED  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching virtual LEDs

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindVirtualLEDList](#_获取虚拟LED列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindVirtualLEDList](#_获取虚拟LED列表) and [NETDEV\_XW\_FindNextVirtualLED](#_逐个获取虚拟LED信息) interfaces.
* Call the [NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息) interface to release resource after virtual LED info is obtained.

**See also:**

[NETDEV\_XW\_FindVirtualLEDList](#_获取虚拟LED列表), [NETDEV\_XW\_FindNextVirtualLED](#_逐个获取虚拟LED信息)

### Create a window

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_CreateWnd  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_SCENE\_WND\_INFO\_S](#_窗口信息结构体) pstWndInfo,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Create a window

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstWndInfo | INOUT | Window info. The window ID is assigned and output by the device. |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Create windows in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_CreateBatchWnd  (  LPVOID lpUserID,  INT32 dwTimeOut,  UINT32 udwTVWallID,  [LPNETDEV\_XW\_WND\_INFO\_LIST\_S](#_窗口列表信息结构体) pstWndInfo,  [LPNETDEV\_XW\_BATCH\_RESULT\_LIST\_S](#_批量操作结果信息结构体) pstResultWndInfo  ); |

**Interface description:**

Create windows in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwTimeOut | IN | Timeout value (second). Range: 10-60 |
| udwTVWallID | IN | Video wall ID |
| pstWndInfo | IN | Window info structure. Window ID is returned after window is created. |
| pstResultWndInfo | OUT | Returned batch operation results |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Modify a window

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_ModifyWnd  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_SCENE\_WND\_INFO\_S](#_窗口信息结构体) pstWndInfo,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Modify a window

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstWndInfo | INOUT | Window info. Window ID is required. |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Modify windows in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_ModifyBatchWnd  (  LPVOID lpUserID,  UINT32 udwTVWallID,  [[LPNETDEV\_XW\_WND\_INFO\_LIST\_S](#_窗口列表信息结构体)](#_窗口列表信息结构体) pstWndInfo,  [LPNETDEV\_XW\_BATCH\_RESULT\_LIST\_S](#_批量操作结果信息结构体) pstResultWndInfo  ); |

**Interface description:**

Modify windows in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTVWallID | IN | Video wall ID |
| pstWndInfo | IN | Window info |
| pstResultWndInfo | OUT | Returned batch operation results |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Delete a window

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteWnd  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwWndID,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Delete a window

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| udwWndID | IN | Window ID |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Magnify/restore a window

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_ZoomScreen  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwWndID,  [LPNETDEV\_XW\_SCREEN\_ZOOM\_INFO\_S](#_屏幕缩放信息结构体) pstScreenZoomInfo  ); |

**Interface description:**

Magnify/restore a window;

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| udwWndID | IN | Window ID |
| pstScreenZoomInfo | IN | Screen zoom info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Delete windows in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteBatchWnd  (  LPVOID lpUserID,  INT32 dwTimeOut,  UINT32 udwTVWallID,  [LPNETDEV\_XW\_BATCH\_RESULT\_LIST\_S](#_批量操作结果信息结构体) pstDeleteWndInfo  ); |

**Interface description:**

Delete windows in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwTimeOut | IN | Timeout value (second). Range: 10-60 |
| udwTVWallID | IN | Video wall ID |
| pstDeleteWndInfo | INOUT | Window info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get window configuration list

#### Get configuration of all windows in the scene

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindWndList  (  LPVOID lpUserID,  UINT32 udwTvWallID  ); |

**Interface description:**

Get configuration of all windows in the scene

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_XW\_FindNextWnd](#_逐个获取查找到的窗口配置信息) and [NETDEV\_XW\_FindCloseWnd](#_关闭查找窗口配置).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindNextWnd](#_逐个获取查找到的窗口配置信息) and [NETDEV\_XW\_FindCloseWnd](#_关闭查找窗口配置) interfaces.
* After the interface is called successfully, call the [NETDEV\_XW\_FindNextWnd](#_逐个获取查找到的窗口配置信息) interface repeatedly to get the next window info.
* Call the [NETDEV\_XW\_FindCloseWnd](#_关闭查找窗口配置) interface to release resource after window info is obtained.

**See also:**

[NETDEV\_XW\_FindNextWnd](#_逐个获取查找到的窗口配置信息), [NETDEV\_XW\_FindCloseWnd](#_关闭查找窗口配置)

#### Get configuration of windows one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextWnd  (  LPVOID lpFindHandle,  [LPNETDEV\_XW\_SCENE\_WND\_INFO\_S](#_窗口信息结构体) pstWndInfo  ); |

**Interface description:**

Get configuration of windows one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindWndList](#_获取场景下所有窗口配置信息) interface. |
| pstWndInfo | OUT | Window configuration |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindWndList](#_获取场景下所有窗口配置信息) and [NETDEV\_XW\_FindCloseWnd](#_关闭查找窗口配置) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_XW\_FindCloseWnd](#_关闭查找窗口配置) interface to release resource after window info is obtained.

**See also:**

[NETDEV\_XW\_FindWndList](#_获取场景下所有窗口配置信息), [NETDEV\_XW\_FindCloseWnd](#_关闭查找窗口配置)

#### Stop searching window configuration

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseWnd  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching window configuration info and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindWndList](#_获取场景下所有窗口配置信息) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindWndList](#_获取场景下所有窗口配置信息) and [[NETDEV\_XW\_FindNextWnd](#_逐个获取查找到的窗口配置信息)](#_逐个获取查找到的窗口配置信息) interfaces.
* Call the [NETDEV\_XW\_FindCloseWnd](#_关闭查找窗口配置) interface to release resource after window info is obtained.

**See also:**

[NETDEV\_XW\_FindWndList](#_获取场景下所有窗口配置信息), [[NETDEV\_XW\_FindNextWnd](#_逐个获取查找到的窗口配置信息)](#_逐个获取查找到的窗口配置信息)

### Get configuration of a window

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetWnd  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_SCENE\_WND\_INFO\_S](#_窗口信息结构体) pstWndInfo  ); |

**Interface description:**

Get configuration of a window

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstWndInfo | INOUT | Window info. Window ID is required. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get binding info of single live video service

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetLiveBind  (  LPVOID lpUserID,  [LPNETDEV\_XW\_DISPLAYER\_ID\_S](#_解码设备ID信息结构体) pstDisplayerID,  [LPNETDEV\_XW\_VIDEO\_SOURCE\_S](#_视频源信息结构体) pstVideoSource  ); |

**Interface description:**

Get binding info of single live video service

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstDisplayerID | IN | Video wall ID, window ID, split window ID |
| pstVideoSource | OUT | Camera service info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Bind a live view service

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_CreateLiveBind  (  LPVOID lpUserID,  [LPNETDEV\_XW\_DISPLAYER\_ID\_S](#_解码设备ID信息结构体) pstDisplayerID,  [LPNETDEV\_XW\_VIDEO\_SOURCE\_S](#_视频源信息结构体) pstVideoSource,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Bind a live view service

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstDisplayerID | IN | Video wall ID, window ID, split window ID |
| pstVideoSource | IN | Camera service info |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Unbind a live view service

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteLiveBind  (  LPVOID lpUserID,  [LPNETDEV\_XW\_DISPLAYER\_ID\_S](#_解码设备ID信息结构体) pstDisplayerID,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Unbind a live view service

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstDisplayerID | IN | Video wall ID, window ID, split window ID |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get live view binding info in the scene

#### Get live view binding info in the scene

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindLiveBindList  (  LPVOID lpUserID,  UINT32 udwTvWallID  ); |

**Interface description:**

Get live view binding info in the scene

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_XW\_FindNextLiveBindInfo](#_逐个获取实况业务绑定信息) and [NETDEV\_XW\_FindCloseLiveBindInfo](#_关闭获取实况业务绑定信息).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindNextLiveBindInfo](#_逐个获取实况业务绑定信息) and [NETDEV\_XW\_FindCloseLiveBindInfo](#_关闭获取实况业务绑定信息) interfaces.
* After the interface is called successfully, call the [NETDEV\_XW\_FindNextLiveBindInfo](#_逐个获取实况业务绑定信息) interface repeatedly to get the next binding info.
* Call the [NETDEV\_XW\_FindCloseLiveBindInfo](#_关闭获取实况业务绑定信息) interface to release resource after live view binding info is obtained.

**See also:**

[NETDEV\_XW\_FindNextLiveBindInfo](#_逐个获取实况业务绑定信息), [NETDEV\_XW\_FindCloseLiveBindInfo](#_关闭获取实况业务绑定信息)

#### Get live view binding info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextLiveBindInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_XW\_VIDEO\_SOURCE\_S](#_视频源信息结构体) pstLiveBindSource  ); |

**Interface description:**

Get live view binding info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindLiveBindList](#_获取场景下的实况业务绑定信息) interface. |
| pstLiveBindSource | OUT | Live view stream info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindLiveBindList](#_获取场景下的实况业务绑定信息) and [NETDEV\_XW\_FindCloseLiveBindInfo](#_关闭获取实况业务绑定信息) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_XW\_FindCloseLiveBindInfo](#_关闭获取实况业务绑定信息) interface to release resource after live view binding info is obtained.

**See also:**

[NETDEV\_XW\_FindLiveBindList](#_获取场景下的实况业务绑定信息), [NETDEV\_XW\_FindCloseLiveBindInfo](#_关闭获取实况业务绑定信息)

#### Stop getting live view binding info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseLiveBindInfo  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop getting live view binding info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindLiveBindList](#_获取场景下的实况业务绑定信息) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindLiveBindList](#_获取场景下的实况业务绑定信息) and [NETDEV\_XW\_FindNextLiveBindInfo](#_逐个获取实况业务绑定信息) interfaces.
* Call the [NETDEV\_XW\_FindCloseLiveBindInfo](#_关闭获取实况业务绑定信息) interface to release resource after live view binding info is obtained.

**See also:**

[NETDEV\_XW\_FindLiveBindList](#_获取场景下的实况业务绑定信息), [NETDEV\_XW\_FindNextLiveBindInfo](#_逐个获取实况业务绑定信息)

### Start passive decoding

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StartPassiveDecode  (  LPVOID lpUserID,  [LPNETDEV\_XW\_DISPLAYER\_ID\_S](#_解码设备ID信息结构体) pstDisplayerID,  [LPNETDEV\_XW\_VIDEO\_SOURCE\_S](#_视频源信息结构体) pstVideoSource,  UINT32 \*pudwTaskNo  ); |

**Interface description:**

Start passive decoding

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstDisplayerID | IN | Device ID info |
| pstVideoSource | IN | Video source info |
| pudwTaskNo | OUT | Service ID, used to control passive decoding |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_StopPassiveDecode](#_停止被动解码), [NETDEV\_SendPassiveData](#_向被动解码通道发送数据)

### Send data to passive decoding channel

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SendPassiveData  (  LPVOID lpUserID,  UINT32 udwTaskNo,  [LPNETDEV\_PASSIVE\_SEND\_S](#_被动解码发送控制参数信息结构体) pstSendData  ); |

**Interface description:**

Send data to passive decoding channel

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTaskNo | IN | Service ID, used to control passive decoding |
| pstSendData | IN | Content of data to send |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_StartPassiveDecode](#_开始被动解码), [NETDEV\_StopPassiveDecode](#_停止被动解码)

### Stop passive decoding

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopPassiveDecode  (  LPVOID lpUserID,  UINT32 udwTaskNO  ); |

**Interface description:**

Stop passive decoding

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTaskNO | IN | Service ID, used to control passive decoding |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_StartPassiveDecode](#_开始被动解码), [NETDEV\_SendPassiveData](#_向被动解码通道发送数据)

### Service stream info

#### Get all service stream info

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindStreamList  (  LPVOID lpUserID,  UINT32 udwTvWallID  ); |

**Interface description:**

Get all service stream info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_XW\_FindNextStreamInfo](#_逐个获取业务流信息) and [NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindNextStreamInfo](#_逐个获取业务流信息) and [NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息) interfaces.
* After the interface is called successfully, call the [NETDEV\_XW\_FindNextStreamInfo](#_逐个获取业务流信息) interface repeatedly to get the next stream info.
* Call the [NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息) interface to release resource after stream info is obtained.

**See also:**

[NETDEV\_XW\_FindNextStreamInfo](#_逐个获取业务流信息), [NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息)

#### Get stream info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextStreamInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_XW\_STREAM\_INFO\_S](#_流业务的详细信息结构体) pstXWStreamInfo  ); |

**Interface description:**

Get stream info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindStreamList](#_获取所有业务流信息) interface. |
| pstXWStreamInfo | OUT | Service stream info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindStreamList](#_获取所有业务流信息) and [NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息) interface to release resource after stream info is obtained.

**See also:**

[NETDEV\_XW\_FindStreamList](#_获取所有业务流信息), [NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息)

#### Stop getting service stream info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseStreamInfo  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop getting service stream info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindStreamList](#_获取所有业务流信息) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindStreamList](#_获取所有业务流信息) and [NETDEV\_XW\_FindNextStreamInfo](#_逐个获取业务流信息) interfaces.
* Call the [NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息) interface to release resource after stream info is obtained.

**See also:**

[NETDEV\_XW\_FindStreamList](#_获取所有业务流信息), [NETDEV\_XW\_FindNextStreamInfo](#_逐个获取业务流信息)

### Get all sequence resources of device

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetSequenceResList  (  LPVOID lpUserID,  UINT32\* udwResNum,  [LPNETDEV\_XW\_SEQUENCE\_RES\_BASE\_INFO\_S](#_轮巡资源基本信息结构体) pstSequenceResInfoList  ); |

**Interface description:**

Get all sequence resources of device

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwResNum | INOUT | Number of sequence resources. When used as an input parameter, it is used to specify the size of the pstSequenceResInfoList array. |
| pstSequenceResInfoList | OUT | Pointer to sequence resource list. Memory is assigned by user. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* udwResNum returns the actual number of sequence resources obtained. If the interface calling failed and the error code is NETDEV\_E\_NEEDMOREDATA, it means the user-assigned memory is not enough.

### Add a sequence resource to device

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_AddSequenceResource  (  LPVOID lpUserID,  [LPNETDEV\_XW\_SEQUENCE\_RES\_DETAIL\_INFO\_S](#_轮巡资源详细信息结构体) pstSeqResInfo,  OUT UINT32 \*pudwLastChange  ); |

**Interface description:**

Add a sequence resource to device

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstSeqResInfo | INOUT | Sequence resource info |
| pudwLastChange | OUT | Summary |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Modify a sequence resource

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_ModifySequenceResource  (  LPVOID lpUserID,  [LPNETDEV\_XW\_SEQUENCE\_RES\_DETAIL\_INFO\_S](#_轮巡资源详细信息结构体) pstSeqResInfo,  UINT32\* pudwLastChange  ); |

**Interface description:**

Modify a sequence resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstSeqResInfo | IN | Sequence resource info |
| pudwLastChange | OUT | Summary |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Delete a sequence resource

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteSequenceResource  (  LPVOID lpUserID,  UINT32 udwSeqResID,  UINT32\* pudwLastChange  ); |

**Interface description:**

Delete a sequence resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwSeqResID | IN | Sequence resource ID |
| pudwLastChange | OUT | Summary |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get detailed info about a sequence resource

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetSequenceResource  (  LPVOID lpUserID,  [LPNETDEV\_XW\_SEQUENCE\_RES\_DETAIL\_INFO\_S](#_轮巡资源详细信息结构体) pstSeqResInfo  ); |

**Interface description:**

Get detailed info about a sequence resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstSeqResInfo | INOUT | Sequence resource info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Create a scene

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_CreateSceneInfo  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_SCENE\_INFO\_BASE\_S](#_场景基本信息结构体) pstSceneBase,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Create a scene

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstSceneBase | INOUT | Input scene name, output scene ID, resource ID (OrderNo) |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Modify a scene

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_ModifySceneInfo  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_SCENE\_MODIFY\_INFO\_S](#_场景修改信息结构体) pstSceneModifyInfo,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Modify a scene

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstSceneModifyInfo | INOUT | Modification info |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Delete a scene

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteSceneInfo  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwSceneID,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Delete a scene

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| udwSceneID | IN | Scene ID |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get the associated scenes of the current scene

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetCurrentSceneID  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 \*pudwSceneID  ); |

**Interface description:**

Get the associated scenes of the current scene

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pudwSceneID | OUT | Scene ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Switch scene

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_SetCurrentSceneID  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwSceneID,  UINT32 \*pudwLastChange  ); |

**Interface description:**

Switch to the current scene

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| udwSceneID | IN | Scene ID |
| pudwLastChange | OUT | Summary, generated by the device, will be updated when configurations change. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get info about the current scene

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetCurrentSceneInfo  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_SCENE\_INFO\_DETAIL\_S](#_场景详细信息结构体) pstSceneDetailInfo  ); |

**Interface description:**

Get info about the current scene

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstSceneDetailInfo | INOUT | Scene info. szTVWallCode is required if video wall code exists. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get scene info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetSceneInfo  (  LPVOID lpUserID,  UINT32 udwSceneID,  [LPNETDEV\_XW\_SCENE\_INFO\_DETAIL\_S](#_场景详细信息结构体) pstSceneDetailInfo  ); |

**Interface description:**

Get scene info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwSceneID | IN | Scene ID |
| pstSceneDetailInfo | INOUT | Scene info. szTVWallCode is required if video wall code exists. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get scene configuration list

#### Get scene configuration list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindSceneCfgList  (  LPVOID lpUserID,  UINT32 udwTvWallID  ); |

**Interface description:**

Get scene configuration list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_XW\_FindNextSceneCfg](#_逐个获取查找到的场景配置信息) and [NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindNextSceneCfg](#_逐个获取查找到的场景配置信息) and [NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景) interfaces.
* After the interface is called successfully, call the [NETDEV\_XW\_FindNextSceneCfg](#_逐个获取查找到的场景配置信息) interface repeatedly to get the next scene info.
* Call the [NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景) interface to release resource after scene info is obtained.

**See also:**

[NETDEV\_XW\_FindNextSceneCfg](#_逐个获取查找到的场景配置信息), [NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景)

#### Get configuration of scenes one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextSceneCfg  (  LPVOID lpFindHandle,  LPNETDEV\_XW\_SCENE\_INFO\_BASIC\_S pstSceneShortInfo  ); |

**Interface description:**

Get configuration of scenes one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindSceneCfgList](#_获取场景配置列表) interface. |
| pstSceneShortInfo | OUT | Scene info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindSceneCfgList](#_获取场景配置列表) and [NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景) interface to release resource after scene info is obtained.

**See also:**

[NETDEV\_XW\_FindSceneCfgList](#_获取场景配置列表), [NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景)

#### Stop searching scenes

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseSceneCfg  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching scene info and release resource.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindSceneCfgList](#_获取场景配置列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindSceneCfgList](#_获取场景配置列表) and [NETDEV\_XW\_FindNextSceneCfg](#_逐个获取查找到的场景配置信息) interfaces.
* Call the [NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景) interface to release resource after scene info is obtained.

**See also:**

[NETDEV\_XW\_FindSceneCfgList](#_获取场景配置列表), [NETDEV\_XW\_FindNextSceneCfg](#_逐个获取查找到的场景配置信息)

### Bind sequence resource to the scene

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_BindSequenceResToScene  (  LPVOID lpUserID,  UINT32 udwTVWallID,  [LPNETDEV\_XW\_SEQUENCE\_RES\_BIND\_INFO\_S](#_轮巡资源绑定信息结构体) pstSeqResBindInfo,  UINT32\* pudwLastChange  ); |

**Interface description:**

Bind sequence resource to the scene

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTVWallID | IN | Video wall ID |
| pstSeqResBindInfo | IN | Sequence resource binding info |
| pudwLastChange | OUT | Summary |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Modify the sequence resource bound to the scene

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_ModifySequenceResToScene  (  LPVOID lpUserID,  UINT32 udwTVWallID,  [LPNETDEV\_XW\_SEQUENCE\_RES\_BIND\_INFO\_S](#_轮巡资源绑定信息结构体) pstSeqResBindInfo,  UINT32\* pudwLastChange  ); |

**Interface description:**

Modify the sequence resource bound to the scene

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTVWallID | IN | Video wall ID |
| pstSeqResBindInfo | IN | Sequence resource binding info |
| pudwLastChange | OUT | Summary |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Delete the sequence resource bound to the scene

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteSequenceResToScene  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwSeqResBindID,  UINT32\* pudwLastChange  ); |

**Interface description:**

Delete the sequence resource bound to the scene

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| udwSeqResBindID | IN | Sequence resource binding ID |
| pudwLastChange | OUT | Summary |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Control sequence playing

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_SequencePlayControl  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_INFO\_S](#_轮巡播放控制信息结构体) pstSeqPlayCtrlInfo  ); |

**Interface description:**

Control sequence playing

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstSeqPlayCtrlInfo | IN | Info about sequence playing control |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Add a scene sequence plan

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_AddSencesSequencePlan  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_SCENES\_PLAN\_INFO\_S](#_场景轮巡计划基本信息结构体) pstScenesPlanInfo,  UINT32\* pudwLastChange  ); |

**Interface description:**

Add a scene sequence plan

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstScenesPlanInfo | INOUT | Pointer to scene sequence plan info. Output resource ID, resource sequence number (OrderNo) |
| pudwLastChange | OUT | Summary |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Modify a scene sequence plan

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_ModifySencesSequencePlan  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_SCENES\_PLAN\_INFO\_S](#_场景轮巡计划基本信息结构体) pstScenesPlanInfo,  UINT32\* pudwLastChange  ); |

**Interface description:**

Modify a scene sequence plan

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstScenesPlanInfo | IN | Scene sequence plan info |
| pudwLastChange | OUT | Summary |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Delete a scene sequence plan

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteSencesSequencePlan  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwScenesSeqPlanID,  UINT32\* pudwLastChange  ); |

**Interface description:**

Delete a scene sequence plan

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| udwScenesSeqPlanID | IN | Scene sequence plan ID |
| pudwLastChange | OUT | Summary |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get a scene sequence plan

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetSencesSequencePlan  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_SCENES\_PLAN\_INFO\_S](#_场景轮巡计划基本信息结构体) pstScenesPlanInfo  ); |

**Interface description:**

Get a scene sequence plan

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstScenesPlanInfo | INOUT | Scene sequence plan info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get scene sequence plan list

#### Get scene sequence plan list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindSencesSequencePlanList  (  LPVOID lpUserID,  UINT32 udwTvWallID  ); |

**Interface description:**

Get scene sequence plan list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_XW\_FindNextSencesSequencePlan](#_逐个获取场景轮巡计划) and [NETDEV\_XW\_FindCloseSencesSequencePlan](#_释放获取场景轮巡计划时内存资源).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindNextSencesSequencePlan](#_逐个获取场景轮巡计划) and [NETDEV\_XW\_FindCloseSencesSequencePlan](#_释放获取场景轮巡计划时内存资源) interfaces.
* After the interface is called successfully, call the [NETDEV\_XW\_FindNextSencesSequencePlan](#_逐个获取场景轮巡计划) interface repeatedly to get the next plan info.
* Call the [NETDEV\_XW\_FindCloseSencesSequencePlan](#_释放获取场景轮巡计划时内存资源) interface to release resource after scene sequence plan info is obtained.

**See also:**

[NETDEV\_XW\_FindNextSencesSequencePlan](#_逐个获取场景轮巡计划), [NETDEV\_XW\_FindCloseSencesSequencePlan](#_释放获取场景轮巡计划时内存资源)

#### Get scene sequence plan one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextSencesSequencePlan  (  LPVOID lpFindHandle,  [LPNETDEV\_XW\_SCENES\_PLAN\_INFO\_S](#_场景轮巡计划基本信息结构体) pstScenesPlanInfo  ); |

**Interface description:**

Get scene sequence plan one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindSencesSequencePlanList](#_获取场景轮巡计划列表) interface. |
| pstScenesPlanInfo | INOUT | Scene sequence plan info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindSencesSequencePlanList](#_获取场景轮巡计划列表) and [NETDEV\_XW\_FindCloseSencesSequencePlan](#_释放获取场景轮巡计划时内存资源) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_XW\_FindCloseSencesSequencePlan](#_释放获取场景轮巡计划时内存资源) interface to release resource after scene sequence plan info is obtained.

**See also:**

[NETDEV\_XW\_FindSencesSequencePlanList](#_获取场景轮巡计划列表), [NETDEV\_XW\_FindCloseSencesSequencePlan](#_释放获取场景轮巡计划时内存资源)

#### Release memory resources used to get scene sequence plans

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseSencesSequencePlan  (  LPVOID lpFindHandle  ); |

**Interface description:**

Release memory resources used to get scene sequence plans

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_XW\_FindSencesSequencePlanList](#_获取场景轮巡计划列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_XW\_FindSencesSequencePlanList](#_获取场景轮巡计划列表) and [NETDEV\_XW\_FindNextSencesSequencePlan](#_逐个获取场景轮巡计划) interfaces.
* Call the [NETDEV\_XW\_FindCloseSencesSequencePlan](#_释放获取场景轮巡计划时内存资源) interface to release resource after scene sequence plan info is obtained.

**See also:**

[NETDEV\_XW\_FindSencesSequencePlanList](#_获取场景轮巡计划列表), [NETDEV\_XW\_FindNextSencesSequencePlan](#_逐个获取场景轮巡计划)

### Control the playing of scene sequence plans

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_ScenesSeqPlanPlayControl  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_SCENES\_PLAN\_PLAY\_CTRL\_INFO\_S](#_场景轮巡计划播放控制信息结构体) pstScenesPlanPlayCtrlInfo  ); |

**Interface description:**

Control the playing of scene sequence plans

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTvWallID | IN | Video wall ID |
| pstScenesPlanPlayCtrlInfo | IN | Info about sequence playing control |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

## Smart Function

### Get system image info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetSystemPicture  (  LPVOID lpUserID,  CHAR\* pszURL,  UINT32 udwSize,  CHAR\* pszdata  ); |

**Interface description:**

Get system image info

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pszURL | IN | Image URL |
| udwSize | IN | Data size before encryption |
| pszdata | OUT | Image data. Memory needs to be allocated dynamically. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Manual linkage operations

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ManualLink  (  LPVOID lpUserID,  UINT32 udwChannelID,  [LPNETDEV\_MANUAL\_LINK\_S](#_手动联动数据结构体) pstManualLink  ); |

**Interface description:**

Manual linkage operations

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwChannelID | IN | Channel ID |
| pstManualLink | IN | Manual linkage data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Register the callback function to receive plate recognition data

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetCarPlateCallBack  (  LPVOID lpUserID,  NETDEV\_CarPlateCallBack\_PF cbCarPlateCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive plate recognition data

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbCarPlateCallBack | IN | Callback function to register |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL \*NETDEV\_CarPlateCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_TMS\_CAR\_PLATE\_INFO\_S](#_车牌识别报文数据信息结构体) pstCarPlateData,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstCarPlateData | IN | Plate recognition data |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Register the callback function to receive face recognition alarms

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetFaceAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_FaceAlarmMessCallBack\_PF cbFaceAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive face recognition alarms

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbFaceAlarmMessCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_FaceAlarmMessCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_FACE\_RECORD\_SNAPSHOT\_INFO\_S](#_人脸识别记录结构体) pstFaceAlarmReportInfo,  LPVOID lpBuf,  INT32 dwBufLen,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstFaceAlarmReportInfo | IN | Reported info |
| lpBuf | IN | Buffer pointer |
| dwBufLen | IN | Length of alarm info structure |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Register the callback function to receive captured face images

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetFaceSnapshotCallBack  (  LPVOID lpUserID,  NETDEV\_FaceSnapshotCallBack\_PF cbFaceSnapshotCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive captured face images

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbFaceSnapshotCallBack | IN | Callback function to register |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL \*NETDEV\_FaceSnapshotCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_TMS\_FACE\_SNAPSHOT\_PIC\_INFO\_S](#_人脸抓拍信息结构体) pstFaceSnapShotData,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstFaceSnapShotData | IN | Face snapshot data |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Register the callback function to receive heatmap data

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetHeatMapCallBack  (  LPVOID lpUserID,  NETDEV\_HeatMapCallBack\_PF cbHeatMapCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive heatmap data

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbHeatMapCallBack | IN | Callback function to register |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL \*NETDEV\_HeatMapCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_TMS\_HEAT\_MAP\_INFO\_S](#_热度图报文数据信息结构体) pstHeatMapData,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstHeatMapData | IN | Heatmap data |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Register the callback function to receive face alarms

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPersonAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_PersonAlarmMessCallBack\_PF cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive face alarms

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbAlarmMessCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_PersonAlarmMessCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_PERSON\_EVENT\_INFO\_S](#_人员报警信息) pstAlarmData,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstAlarmData | IN | Alarm data |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Register the callback function to receive structured alarms

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetStructAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_StructAlarmMessCallBack\_PF cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive structured alarms

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbAlarmMessCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_StructAlarmMessCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_STRUCT\_ALARM\_INFO\_S](#_结构化告警上报信息) pstAlarmInfo,  [LPNETDEV\_STRUCT\_DATA\_INFO\_S](#_结构化数据信息) pstAlarmData,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstAlarmInfo | IN | Alarm info |
| pstAlarmData | IN | Alarm data |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Register the callback function to receive plate recognition alarms

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetVehicleAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_VehicleAlarmMessCallBack\_PF cbVehicleAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive plate recognition alarms

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbVehicleAlarmMessCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_VehicleAlarmMessCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_VEH\_RECOGNITION\_EVENT\_S](#_车辆识别事件结构体) pstVehicleAlarmInfo,  LPVOID lpBuf,  INT32 dwBufLen,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstVehicleAlarmInfo | IN | Reported info |
| lpBuf | IN | Buffer pointer |
| dwBufLen | IN | Length of alarm info structure |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Subscribe to VCA events

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SubscribeSmart  (  LPVOID lpUserID,  [LPNETDEV\_SUBSCRIBE\_SMART\_INFO\_S](#_订阅智能事件信息结构体) pstSubscribeInfo,  [LPNETDEV\_SMART\_INFO\_S](#_智能事件信息结构体) pstSmartInfo  ); |

**Interface description:**

Subscribe to VCA events

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstSubscribeInfo | IN | Subscription info |
| pstSmartInfo | INOUT | Smart event info. Subscription ID will be returned if successful. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Call [NETDEV\_SetAlarmCallBack\_V30](#_注册回调函数,接收报警消息等) to register callback function before subscription.

**See also:**

[NETDEV\_UnsubscribeSmart](#_取消订阅智能事件)

### Cancel VCA event subscription

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_UnsubscribeSmart  (  LPVOID lpUserID,  [LPNETDEV\_SMART\_INFO\_S](#_智能事件信息结构体) pstSmartInfo  ); |

**Interface description:**

Cancel VCA event subscription

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstSmartInfo | IN | VCA event |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SubscribeSmart](#_订阅智能事件)

### Alarm subscription via LAPI

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SubscibeLapiAlarm  (  LPVOID lpUserID,  [LPNETDEV\_LAPI\_SUB\_INFO\_S](#_Lapi告警订阅信息结构体) pstSubInfo,  [LPNETDEV\_SUBSCRIBE\_SUCC\_INFO\_S](#_订阅信息成功返回信息结构体) pstSubSuccInfo  ); |

**Interface description:**

Alarm subscription via LAPI

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstSubInfo | IN | Alarm subscription request |
| pstSubSuccInfo | OUT | Returned info after successful subscription |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_UnSubLapiAlarm](#_取消LAPI告警订阅)

### Cancel alarm subscription via LAPI

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_UnSubLapiAlarm  (  LPVOID lpUserID,  UINT32 udwID  ); |

**Interface description:**

Cancel alarm subscription via LAPI

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwID | IN | Alarm subscription ID, which is obtained by calling the [NETDEV\_SubscibeLapiAlarm](#_LAPI告警订阅) interface |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

**See also:**

[NETDEV\_SubscibeLapiAlarm](#_LAPI告警订阅)

### Face recognition record

#### Get face recognition record

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindFaceRecordDetailList  (  LPVOID lpUserID,  [LPNETDEV\_ALARM\_LOG\_COND\_LIST\_S](#_告警日志查询条件列表) pstFindCond,  [LPNETDEV\_SMART\_ALARM\_LOG\_RESULT\_INFO\_S](#_告警记录返回信息（人脸识别和车牌识别）结构体) pstResultInfo  ); |

**Interface description:**

Get face recognition record

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstFindCond | IN | Search criteria |
| pstResultInfo | OUT | Face recognition record info |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextFaceRecordDetail](#_逐个获取人脸识别记录) and [[NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录)](#_关闭查找人脸识别记录).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextFaceRecordDetail](#_逐个获取人脸识别记录) and [NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextFaceRecordDetail](#_逐个获取人脸识别记录) interface repeatedly to get the next record.
* Call the [NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录) interface to release resource and stop the search after log info is obtained.

**See also:**

[NETDEV\_FindNextFaceRecordDetail](#_逐个获取人脸识别记录), [NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录)

#### Get face recognition record one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextFaceRecordDetail  (  LPVOID lpFindHandle,  [LPNETDEV\_FACE\_RECORD\_SNAPSHOT\_INFO\_S](#_人脸识别记录结构体) pstRecordInfo  ); |

**Interface description:**

Get face recognition record one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录) interface. |
| pstRecordInfo | OUT | Pointer to face recognition record info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录) and [NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录) interface to release resource and stop the search after log info is obtained.

**See also:**

[NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录), [NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录)

#### Stop getting face recognition record

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseFaceRecordDetail  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching face recognition record and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录) and [NETDEV\_FindNextFaceRecordDetail](#_逐个获取人脸识别记录) interfaces.
* Call the [NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录) interface to release resource and stop the search after log info is obtained.

**See also:**

[NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录), [NETDEV\_FindNextFaceRecordDetail](#_逐个获取人脸识别记录)

#### Get face image info of a face recognition record

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetFaceRecordImageInfo  (  LPVOID lpUserID,  UINT32 udwRecordID,  UINT32 udwFaceImageType,  [LPNETDEV\_FILE\_INFO\_S](#_文件信息结构体) pstFileInfo  ); |

**Interface description:**

Get face image info of a face recognition record

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwRecordID | IN | Face recognition alarm record ID |
| udwFaceImageType | IN | Pass-through record type |
| pstFileInfo | INOUT | Face image info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Parking lot

#### Get all entrances/exits of a parking lot

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetParkEntranceInfoList  (  LPVOID lpUserID,  UINT32 udwParkID,  [LPNETDEV\_PARK\_ENTRANCE\_INFO\_LIST\_S](#_停车场出入口信息列表结构体) pstParkEntranceInfoList  ); |

**Interface description:**

Get all entrances/exits of a parking lot

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwParkID | IN | Parking lot ID |
| pstParkEntranceInfoList | OUT | Entrance/exit list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Get parking lots in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetParkInfoList  (  LPVOID lpUserID,  [LPNETDEV\_PARK\_INFO\_LIST\_S](#_停车场信息列表结构体) pstParkInfoList  ); |

**Interface description:**

Get parking lots in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstParkInfoList | OUT | Parking lot list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Get all entrance/exit lanes of a parking lot

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetParkLaneInfoList  (  LPVOID lpUserID,  [LPNETDEV\_PARK\_ENTRANCE\_ID\_S](#_停车场出入口ID结构体) pstParkEntranceID,  [LPNETDEV\_VEH\_LANE\_INFO\_LIST\_S](#_车道信息列表结构体) pstVehicleLaneList  ); |

**Interface description:**

Get all entrance/exit lanes of a parking lot

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstParkEntranceID | IN | Entrance/exit ID |
| pstVehicleLaneList | OUT | Entrance/exit list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Issue third-party payment order

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_IssuePayOrder  (  LPVOID lpUserID,  [LPNETDEV\_PARK\_PAY\_ORDER\_INFO\_S](#_停车场支付订单信息结构体) pstPayOrder,  [LPNETDEV\_PARK\_COST\_S](#_停车花费结构体) pstCost  ); |

**Interface description:**

Issue third-party payment order

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstPayOrder | IN | Order info |
| pstCost | OUT | Parking cost info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Register the callback function to receive parking lot events

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetParkEventCallBack  (  LPVOID lpUserID,  NETDEV\_ParkEventMessCallBack\_PF cbParkAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive parking lot events

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbParkAlarmMessCallBack | IN | Callback function for data receiving |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_ParkEventMessCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_PARK\_EVENT\_INFO\_S](#_停车场事件信息结构体) stParkAlarmInfo,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| stParkAlarmInfo | IN | Reported info |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Time template

#### Add time template

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddTimeTemplate  (  LPVOID lpUserID,  [LPNETDEV\_SYSTEM\_TIME\_TEMPLATE\_S](#_时间模板配置结构体) pstTimeTemplate,  UINT32 \*pudwTemplateID  ); |

**Interface description:**

Add time template

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstTimeTemplate | IN | Time template info |
| pudwTemplateID | OUT | Time template ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* VMS does not support specifying template ID.
* If the template ID passed in is not 0, then the template ID will be used as the index and returned. If the template ID passed in is an existing value, then the interface returns failure.
* If the template ID passed in is 0, then a unique ID will be assigned by the terminal and then returned.

#### Delete time templates in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_BatchDeleteTimeTemplate  (  LPVOID lpUserID,  [LPNETDEV\_ID\_LIST\_S](#_通用ID列表结构体) pstTemplateIDList,  [LPNETDEV\_BATCH\_OPERATOR\_LIST\_S](#_删除人脸布控返回的信息列表结构体) pstDeleteResults  ); |

**Interface description:**

Delete time templates in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstTemplateIDList | IN | IDs of deleted templates |
| pstDeleteResults | OUT | Returned batch operation results |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Only VMS supports batch delete.
* pstDeleteResults->udwNum should not be less than pstTemplateIDList->udwNum;
* The output parameter returns udwNum, udwID, udwResultCode;

#### Get time templates in batches

##### Get time templates of a specified template type in batches

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindTimeTemplateByTypeList  (  LPVOID lpUserID,  UINT32 udwTemplateType  ); |

**Interface description:**

Get time templates of a specified template type in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwTemplateType | IN | Time template type. Only applicable to VMS. 0: Recording time template 1: Alarm time template 2: User time template |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextTimeTemplateByTypeInfo](#_逐个获取按类型找到的时间模板信息) and [NETDEV\_FindCloseTimeTemplateByTypeList](#_关闭查找按类型找到的时间模板列表).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextTimeTemplateByTypeInfo](#_逐个获取按类型找到的时间模板信息) and [NETDEV\_FindCloseTimeTemplateByTypeList](#_关闭查找按类型找到的时间模板列表) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextTimeTemplateByTypeInfo](#_逐个获取按类型找到的时间模板信息) interface to get the next template info.
* Call the [NETDEV\_FindCloseTimeTemplateByTypeList](#_关闭查找按类型找到的时间模板列表) interface to release resource and stop the search after log info is obtained.

**See also:**

[NETDEV\_FindNextTimeTemplateByTypeInfo](#_逐个获取按类型找到的时间模板信息), [NETDEV\_FindCloseTimeTemplateByTypeList](#_关闭查找按类型找到的时间模板列表)

##### Get time template info by type one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextTimeTemplateByTypeInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_TIME\_TEMPLATE\_BASE\_INFO\_S](#_时间模板结构体) pstTimeTemplate  ); |

**Interface description:**

Get time template info by type one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_FindTimeTemplateByTypeList](#_批量获取指定模板类型的时间模板信息) interface. |
| pstTimeTemplate | OUT | Time template |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindTimeTemplateByTypeList](#_批量获取指定模板类型的时间模板信息) and [NETDEV\_FindCloseTimeTemplateByTypeList](#_关闭查找按类型找到的时间模板列表) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseTimeTemplateByTypeList](#_关闭查找按类型找到的时间模板列表) interface to release resource and stop the search after log info is obtained.

**See also:**

[NETDEV\_FindTimeTemplateByTypeList](#_批量获取指定模板类型的时间模板信息), [NETDEV\_FindCloseTimeTemplateByTypeList](#_关闭查找按类型找到的时间模板列表)

##### Stop getting time templates by type

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseTimeTemplateByTypeList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching time template list by type and release resource.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_FindTimeTemplateByTypeList](#_批量获取指定模板类型的时间模板信息) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindTimeTemplateByTypeList](#_批量获取指定模板类型的时间模板信息) and [NETDEV\_FindNextTimeTemplateByTypeInfo](#_逐个获取按类型找到的时间模板信息) interfaces.
* Call the [NETDEV\_FindCloseTimeTemplateByTypeList](#_关闭查找按类型找到的时间模板列表) interface to release resource and stop the search after log info is obtained.

**See also:**

[NETDEV\_FindTimeTemplateByTypeList](#_批量获取指定模板类型的时间模板信息), [NETDEV\_FindNextTimeTemplateByTypeInfo](#_逐个获取按类型找到的时间模板信息)

#### Get a time template

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetTimeTemplate  (  LPVOID lpUserID,  [LPNETDEV\_SYSTEM\_TIME\_TEMPLATE\_S](#_时间模板配置结构体) pstTimeTemplate  ); |

**Interface description:**

Get a time template

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstTimeTemplate | INOUT | Time template info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Modify a time template

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyTimeTemplate  (  LPVOID lpUserID,  [LPNETDEV\_SYSTEM\_TIME\_TEMPLATE\_S](#_时间模板配置结构体) pstTimeTemplate  ); |

**Interface description:**

Modify a time template

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstTimeTemplate | IN | Time template info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Access control

#### Person management in access control

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ACSPersonCtrl  (  LPVOID lpUserID,  INT32 dwCommand,  [LPNETDEV\_ACS\_PERSON\_INFO\_S](#_门禁人员信息结构体) pstACSPersonInfo  ); |

**Interface description:**

Person management

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwCommand | IN | Person management command. See [NETDEV\_ACS\_PERSON\_COMMOND\_TYPE\_E](#_门禁人员管理命令枚举) |
| pstACSPersonInfo | INOUT | Person info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Sign out visitor

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ACSPersonSignout  (  LPVOID lpUserID,  UINT32 udwPersonID  ); |

**Interface description:**

Sign out visitor

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwPersonID | IN | Person ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Add visitor blocklist

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddACSPersonBlackList  (  LPVOID lpUserID,  [LPNETDEV\_ACS\_PERSON\_BLACKLIST\_INFO\_S](#_黑名单信息结构体) pstBlackListInfo,  UINT32 \*pUdwBlackListID  ); |

**Interface description:**

Add visitor blocklist

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstBlackListInfo | IN | Blocklist info |
| pUdwBlackListID | OUT | Blocklist ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Add access control card for a person

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddACSPersonCardInfo  (  LPVOID lpUserID,  UINT32 udwACSPersonID,  [LPNETDEV\_ACS\_PERSON\_CARD\_INFO\_S](#_人员所持门禁卡信息结构体) pstACSPersonCardInfo  ); |

**Interface description:**

Add access control card for a person (i.e. to issue card)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwACSPersonID | IN | Person ID |
| pstACSPersonCardInfo | INOUT | Access card info. udwCardID is the output parameter |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Add persons in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddACSPersonList  (  LPVOID lpUserID,  [LPNETDEV\_ACS\_PERSON\_LIST\_S](#_人员列表结构体) pstACSPersonList,  [LPNETDEV\_XW\_BATCH\_RESULT\_LIST\_S](#_批量操作结果信息结构体) pstResultList  ); |

**Interface description:**

Add persons in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstACSPersonList | IN | Person list. The size of a single image is 2M. |
| pstResultList | OUT | Returned person list |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Add permission group

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddACSPersonPermissionGroup  (  LPVOID lpUserID,  [LPNETDEV\_ACS\_PERMISSION\_INFO\_S](#_授权信息结构体) pstPermissionGroupInfo,  UINT32 \*pUdwGroupID  ); |

**Interface description:**

Add permission group

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstPermissionGroupInfo | IN | Permission group info |
| pUdwGroupID | OUT | Permission group ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Delete visitor blocklist

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteACSPersonBlackList  (  LPVOID lpUserID,  [LPNETDEV\_OPERATE\_LIST\_S](#_批处理列表) pstBlackList  ); |

**Interface description:**

Delete visitor blocklist

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstBlackList | OUT | Blocklist |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Delete access control card for a person

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteACSPersonCardInfo  (  LPVOID lpUserID,  UINT32 udwACSPersonID  ); |

**Interface description:**

Delete access control card for a person (i.e. to return card)

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwACSPersonID | IN | Person ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Delete persons in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteACSPersonList  (  LPVOID lpUserID,  [LPNETDEV\_FACE\_BATCH\_LIST\_S](#_人脸识别模块批量操作列表结构体) pstBatchCtrlInfo  ); |

**Interface description:**

Delete persons in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstBatchCtrlInfo | INOUT | Batch control info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Delete permission group

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteACSPersonPermissionGroup  (  LPVOID lpUserID,  [LPNETDEV\_OPERATE\_LIST\_S](#_批处理列表) pstPermissionIDList,  [LPNETDEV\_BATCH\_OPERATOR\_LIST\_S](#_删除人脸布控返回的信息列表结构体) pstResutList  ); |

**Interface description:**

Delete permission group

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstPermissionIDList | IN | Permission ID list |
| pstResutList | OUT | Returned operation result |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Control doors in batches

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DoorBatchCtrl  (  LPVOID lpUserID,  INT32 dwCommand,  [LPNETDEV\_OPERATE\_LIST\_S](#_批处理列表) pstBatchCtrlInfo  ); |

**Interface description:**

Control doors in batches

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwCommand | IN | Door control command. See [NETDEV\_DOORCTRL\_ACTION\_TYPE\_E](#_报警点通道控制命令枚举) |
| pstBatchCtrlInfo | INOUT | Batch control info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Control doors

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DoorCtrl  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwCommand  ); |

**Interface description:**

Control doors

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwChannelID | IN | Channel ID |
| dwCommand | IN | Door control command. See [NETDEV\_DOORCTRL\_ACTION\_TYPE\_E](#_报警点通道控制命令枚举) |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Get access record

##### Get access record

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindACSAttendanceLogList  (  LPVOID lpUserID,  [LPNETDEV\_ALARM\_LOG\_COND\_LIST\_S](#_告警日志查询条件列表) pstFindCond,  [LPNETDEV\_BATCH\_OPERATE\_BASIC\_S](#_批量查询返回的基本信息结构体) pstResultInfo  ); |

**Interface description:**

Get access record

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstFindCond | IN | Entry/exit record search criteria |
| pstResultInfo | OUT | Actual total number of access records |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextACSAttendanceLog](#_获取下一条出入记录) and [NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextACSAttendanceLog](#_获取下一条出入记录) and [NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextACSAttendanceLog](#_获取下一条出入记录) interface to get the next record.
* Call the [NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源) interface to release resource and stop the search after access records are obtained.

**See also:**

[NETDEV\_FindNextACSAttendanceLog](#_获取下一条出入记录), [NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源)

##### Get the next access record

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextACSAttendanceLog  (  LPVOID lpFindHandle,  [LPNETDEV\_ACS\_ATTENDANCE\_LOG\_INFO\_S](#_出入记录信息结构体) pstACSLogInfo  ); |

**Interface description:**

Get the next access record

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Handle to access record list, which is the return value from the [NETDEV\_FindACSAttendanceLogList](#_查询出入记录) interface. |
| pstACSLogInfo | OUT | Access record info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindACSAttendanceLogList](#_查询出入记录) and [NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源) interface to release resource after access records are obtained.

**See also:**

[NETDEV\_FindACSAttendanceLogList](#_查询出入记录), [NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源)

##### Close the search handle to access records

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseACSAttendanceLogList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Close the search handle to access records

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Handle to access record list, which is the return value from the [NETDEV\_FindACSAttendanceLogList](#_查询出入记录) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindACSAttendanceLogList](#_查询出入记录) and [NETDEV\_FindNextACSAttendanceLog](#_获取下一条出入记录) interfaces.
* Call the [NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源) interface to release resource after access records are obtained.

**See also:**

[NETDEV\_FindACSAttendanceLogList](#_查询出入记录), [[[NETDEV\_FindNextACSAttendanceLog](#_获取下一条出入记录)](#_获取下一条出入记录)](#_获取下一条出入记录)

#### Get permission group

##### Get permission group list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindACSPermissionGroupList  (  LPVOID lpUserID,  [LPNETDEV\_PERSON\_QUERY\_INFO\_S](#_人员信息查询条件结构体) pstQueryCond,  [LPNETDEV\_BATCH\_OPERATE\_BASIC\_S](#_批量查询返回的基本信息结构体) pstResultInfo  ); |

**Interface description:**

Get permission group list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstQueryCond | IN | Search criteria |
| pstResultInfo | OUT | Returned result |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1) and [NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1) and [NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1) interface repeatedly to get the next group info.
* Call the [NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源) interface to release resource after access control permission group info is obtained.

**See also:**

[NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1), [NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源)

##### Get the next record

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextACSPermissionGroupInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_ACS\_PERMISSION\_INFO\_S](#_授权信息结构体) pstACSPermissionInfo  ); |

**Interface description:**

Get the next permission group record

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Handle to access permission group list, which is the return value from the [NETDEV\_FindACSPermissionGroupList](#_查看门禁授权组列表) interface. |
| pstACSPermissionInfo | OUT | Permission group info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindACSPermissionGroupList](#_查看门禁授权组列表) and [NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源) interface to release resource after access control permission group info is obtained.

**See also:**

[NETDEV\_FindACSPermissionGroupList](#_查看门禁授权组列表), [NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源)

##### Close the search handle to permission groups

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseACSPermissionGroupList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Close the search handle to permission groups

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Handle to access permission group list, which is the return value from the [NETDEV\_FindACSPermissionGroupList](#_查看门禁授权组列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindACSPermissionGroupList](#_查看门禁授权组列表) and [NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1) interfaces.
* Call the [NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源) interface to release resource after access control permission group info is obtained.

**See also:**

[NETDEV\_FindACSPermissionGroupList](#_查看门禁授权组列表), [NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1)

#### Get visitor blocklist

##### Get visitor blocklist

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindACSPersonBlackList  (  LPVOID lpUserID,  [LPNETDEV\_PAGED\_QUERY\_INFO\_S](#_查询条件结构体) pstQueryCond,  [LPNETDEV\_BATCH\_OPERATE\_BASIC\_S](#_批量查询返回的基本信息结构体) pstResultInfo  ); |

**Interface description:**

Get visitor blocklist

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstQueryCond | IN | Search criteria |
| pstResultInfo | OUT | Returned result |

**Return value:**

Null means failure, other values are used as the input parameter of [NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录) and [NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录) and [NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录) interface repeatedly to get the next blocklist info.
* Call the [NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1) interface to release resource after visitor blocklist info is obtained.

**See also:**

[NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录), [NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1)

##### Get the next record

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextACSPersonBlackListInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_ACS\_PERSON\_BLACKLIST\_INFO\_S](#_黑名单信息结构体) pstBlackListInfo  ); |

**Interface description:**

Get the next visitor blacklist record

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle to visitor blocklist info, which is the return value from the [NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表) interface. |
| pstBlackListInfo | OUT | Visitor blocklist |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表) and [NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1) interface to release resource after visitor blocklist info is obtained.

**See also:**

[NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表), [NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1)

##### Close the search handle to visitor blocklist

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseACSPersonBlackList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Close the search handle to visitor blocklist

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle to visitor blocklist info, which is the return value from the [NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表) and [NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录) interfaces.
* Call the [NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1) interface to release resource after visitor blocklist info is obtained.

**See also:**

[NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表), [NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录)

#### Get person info in access control

##### Get person list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindACSPersonList  (  LPVOID lpUserID,  [LPNETDEV\_PERSON\_QUERY\_INFO\_S](#_人员信息查询条件结构体) pstQueryCond,  [LPNETDEV\_BATCH\_OPERATE\_BASIC\_S](#_批量查询返回的基本信息结构体) pstResultInfo  ); |

**Interface description:**

Get person list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstQueryCond | IN | Access control person search criteria |
| pstResultInfo | OUT | Returned info |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextACSPersonInfo](#_获取下一个门禁人员信息) and [NETDEV\_FindCloseACSPersonInfo](#_关闭门禁人员信息列表资源).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextACSPersonInfo](#_获取下一个门禁人员信息) and [NETDEV\_FindCloseACSPersonInfo](#_关闭门禁人员信息列表资源) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextACSPersonInfo](#_获取下一个门禁人员信息) interface repeatedly to get the next person info.
* Call the [NETDEV\_FindCloseACSPersonInfo](#_关闭门禁人员信息列表资源) interface to release resource and stop the search after person info is obtained.

**See also:**

[NETDEV\_FindNextACSPersonInfo](#_获取下一个门禁人员信息), [NETDEV\_FindCloseACSPersonInfo](#_关闭门禁人员信息列表资源)

##### Get info about the next person

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextACSPersonInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_ACS\_PERSON\_BASE\_INFO\_S](#_门禁人员基本信息结构体) pstACSPersonInfo  ); |

**Interface description:**

Get info about the next person

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle to person list, which is the return value from the [NETDEV\_FindACSPersonList](#_获取门禁人员信息列表) interface. |
| pstACSPersonInfo | OUT | Person info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindACSPersonList](#_获取门禁人员信息列表) and [NETDEV\_FindCloseACSPersonInfo](#_关闭门禁人员信息列表资源) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseACSPersonInfo](#_关闭门禁人员信息列表资源) interface to release resource and stop the search after person info is obtained.

**See also:**

[NETDEV\_FindACSPersonList](#_获取门禁人员信息列表), [NETDEV\_FindCloseACSPersonInfo](#_关闭门禁人员信息列表资源)

##### Stop searching person list and release resource

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseACSPersonInfo  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching person list and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle to person list, which is the return value from the [NETDEV\_FindACSPersonList](#_获取门禁人员信息列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindACSPersonList](#_获取门禁人员信息列表) and [NETDEV\_FindNextACSPersonInfo](#_获取下一个门禁人员信息) interfaces.
* Call the [NETDEV\_FindCloseACSPersonInfo](#_关闭门禁人员信息列表资源) interface to release resource and stop searching after person info is obtained.

**See also:**

[NETDEV\_FindACSPersonList](#_获取门禁人员信息列表), [NETDEV\_FindNextACSPersonInfo](#_获取下一个门禁人员信息)

#### Get visitor records

##### Get visitor records

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindACSVisitLogList  (  LPVOID lpUserID,  [LPNETDEV\_ALARM\_LOG\_COND\_LIST\_S](#_告警日志查询条件列表) pstFindCond,  [LPNETDEV\_BATCH\_OPERATE\_BASIC\_S](#_批量查询返回的基本信息结构体) pstResultInfo  ); |

**Interface description:**

Get visitor records

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstFindCond | IN | Visitor record search criteria |
| pstResultInfo | OUT | Actual total number of visitor records |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录) and [NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录) and [NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录) interface to get the next log.
* Call the [NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源) interface to release resource after visitor record info is obtained.

**See also:**

[NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录), [NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源)

##### Get the next visitor record

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextACSVisitLog  (  LPVOID lpFindHandle,  [LPNETDEV\_ACS\_VISIT\_LOG\_INFO\_S](#_访客记录信息结构体) pstACSLogInfo  ); |

**Interface description:**

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle to access record list, which is the return value from the [[NETDEV\_FindACSVisitLogList](#_查询访客记录)](#_查询访客记录) interface. |
| pstACSLogInfo | OUT | Access record info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [[NETDEV\_FindACSVisitLogList](#_查询访客记录)](#_查询访客记录) and [NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源) interface to release resource after access records are obtained.

**See also:**

[[NETDEV\_FindACSVisitLogList](#_查询访客记录)](#_查询访客记录), [NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源)

##### Close the search handle to access records

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseACSVisitLog  (  LPVOID lpFindHandle  ); |

**Interface description:**

Close the search handle to access records

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle to access record list, which is the return value from the [NETDEV\_FindACSVisitLogList](#_查询访客记录) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindACSVisitLogList](#_查询访客记录) and [NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录) interfaces.
* Call the [NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源) interface to release resource after access records are obtained.

**See also:**

[NETDEV\_FindACSVisitLogList](#_查询访客记录), [NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录)

#### Get a visitor blocklist

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetACSPersonBlackList  (  LPVOID lpUserID,  [LPNETDEV\_ACS\_PERSON\_BLACKLIST\_INFO\_S](#_黑名单信息结构体) pstBlackListInfo  ); |

**Interface description:**

Get a visitor blocklist

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstBlackListInfo | INOUT | Blocklist info, among which, udwBlackListID is passed in as the input parameter |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Get info about access cards held by a person

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetACSPersonCardInfo  (  LPVOID lpUserID,  UINT32 udwACSPersonID,  [LPNETDEV\_ACS\_PERSON\_CARD\_INFO\_S](#_人员所持门禁卡信息结构体) pstACSPersonCardInfo  ); |

**Interface description:**

Get info about access cards held by a person

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwACSPersonID | IN | Person ID |
| pstACSPersonCardInfo | OUT | Access card info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Get info about permissions assigned to a person

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetACSPersonPermission  (  LPVOID lpUserID,  UINT32 udwPersonID,  [LPNETDEV\_ACS\_DOOR\_PERMISSION\_INFO\_S](#_门授权信息结构体) pstPermissionInfo  ); |

**Interface description:**

Get info about permissions assigned to a person

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwPersonID | IN | Person ID |
| pstPermissionInfo | OUT | Permission info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Get info about a permission group

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetSinglePermGroupInfo  (  LPVOID lpUserID,  UINT32 udwPermissionGroupID,  [LPNETDEV\_ACS\_PERMISSION\_INFO\_S](#_授权信息结构体) pstAcsPerssionInfo  ); |

**Interface description:**

Get info about a permission group

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwPermissionGroupID | IN | Permission group ID |
| pstAcsPerssionInfo | INOUT | Permission group info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Modify visitor blocklist

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyACSPersonBlackList  (  LPVOID lpUserID,  [LPNETDEV\_ACS\_PERSON\_BLACKLIST\_INFO\_S](#_黑名单信息结构体) pstBlackListInfo  ); |

**Interface description:**

Modify visitor blocklist

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstBlackListInfo | IN | Blocklist info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Modify permission group

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyACSPersonPermissionGroup  (  LPVOID lpUserID,  [LPNETDEV\_ACS\_PERMISSION\_INFO\_S](#_授权信息结构体) pstPermissionInfo  ); |

**Interface description:**

Modify permission group

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstPermissionInfo | IN | Permission group info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Set permission for a person

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetACSPersonPermission  (  LPVOID lpUserID,  UINT32 udwPersonID,  [LPNETDEV\_ACS\_DOOR\_PERMISSION\_INFO\_S](#_门授权信息结构体) pstPermissionInfo  ); |

**Interface description:**

Set permission for a person

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwPersonID | IN | Person ID |
| pstPermissionInfo | IN | Permission info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Register the callback function to receive turnstile alarms

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetAlarmFGCallBack  (  LPVOID lpUserID,  NETDEV\_AlarmMessFGCallBack\_PF cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive turnstile alarms

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbAlarmMessCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_AlarmMessFGCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_PERSON\_VERIFICATION\_S](#_人员核验结构体) pstAlarmData,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstAlarmData | IN | Alarm info |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Object search

#### Start object search

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StartObjectSearch  (  LPVOID lpUserID,  [LPNETDEV\_QUERY\_CHN\_CONDITION\_S](#_条件查询通道信息结构体) pstStartInfo,  UINT32\* pudwSearchID  ); |

**Interface description:**

Start object search

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstStartInfo | IN | Search start info |
| pudwSearchID | OUT | Service ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Get object search progress

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetObjectSearchProg  (  LPVOID lpUserID,  UINT32 udwSearchID,  UINT32\* pudwPercent  ); |

**Interface description:**

Get object search progress

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwSearchID | IN | Service ID |
| pudwPercent | OUT | Search progress |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Get object search result

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindObjectSearchList  (  LPVOID lpUserID,  [LPNETDEV\_PIC\_QUERY\_COND\_S](#_图片查询条件) pstQueryCond,  [LPNETDEV\_PIC\_QUERY\_RESULT\_S](#_图片查询结果) pstQueryResult  ); |

**Interface description:**

Get object search result

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstQueryCond | IN | Search criteria |
| pstQueryResult | OUT | Search result |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextObjectSearchInfo](#_逐个获取查找到的目标检索结果) and [NETDEV\_FindCloseObjectSearchList](#_关闭目标检索结果查找,释放资源).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextObjectSearchInfo](#_逐个获取查找到的目标检索结果) and [NETDEV\_FindCloseObjectSearchList](#_关闭目标检索结果查找,释放资源) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextObjectSearchInfo](#_逐个获取查找到的目标检索结果) interface to get the next search result.
* Call the [NETDEV\_FindCloseObjectSearchList](#_关闭目标检索结果查找,释放资源) interface to release resource after object search result is obtained.

#### Get object search result one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextObjectSearchInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_OBJECT\_RESULT\_INFO\_S](#_目标结果信息) pstObjectResultInfo  ); |

**Interface description:**

Get object search result

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_FindObjectSearchList](#_查询目标检索结果) interface. |
| pstObjectResultInfo | IN | Search result |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindObjectSearchList](#_查询目标检索结果) and [NETDEV\_FindCloseObjectSearchList](#_关闭目标检索结果查找,释放资源) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseObjectSearchList](#_关闭目标检索结果查找,释放资源) interface to release resource after object search result is obtained.

#### Stop searching object search result and release resource

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseObjectSearchList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching object search result and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_FindObjectSearchList](#_查询目标检索结果) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindObjectSearchList](#_查询目标检索结果) and [NETDEV\_FindNextObjectSearchInfo](#_逐个获取查找到的目标检索结果) interfaces.
* Call the [NETDEV\_FindCloseObjectSearchList](#_关闭目标检索结果查找,释放资源) interface to release resource after object search result is obtained.

#### Stop object search

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopObjectSearch  (  LPVOID lpUserID,  UINT32 udwSearchID  ); |

**Interface description:**

Stop searching object search result and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwSearchID | IN | Service ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Get data related to an alarm

#### Get alarm related data list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindAlarmRelatedDataList  (  LPVOID lpUserID,  INT32 dwAlarmID  ); |

**Interface description:**

Get alarm related data list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| dwAlarmID | IN | Alarm ID |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextAlarmRelatedDataInfo](#_获取指定告警的关联数据) and [NETDEV\_FindCloseAlarmRelatedDataList](#_关闭查询指定告警的关联数据).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextAlarmRelatedDataInfo](#_获取指定告警的关联数据) and [NETDEV\_FindCloseAlarmRelatedDataList](#_关闭查询指定告警的关联数据) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextAlarmRelatedDataInfo](#_获取指定告警的关联数据) interface repeatedly to get the next data info.
* Call the [NETDEV\_FindCloseAlarmRelatedDataList](#_关闭查询指定告警的关联数据) interface to release resource after alarm related data is obtained.

#### Get alarm related data

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextAlarmRelatedDataInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_ALARM\_RELATED\_DATA\_S](#_告警关联数据) pstAlarmRelatedData  ); |

**Interface description:**

Get alarm related data

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_FindAlarmRelatedDataList](#_获取指定告警的关联数据列表) interface. |
| pstAlarmRelatedData | IN | Alarm related data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindAlarmRelatedDataList](#_获取指定告警的关联数据列表) and [NETDEV\_FindCloseAlarmRelatedDataList](#_关闭查询指定告警的关联数据) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseAlarmRelatedDataList](#_关闭查询指定告警的关联数据) interface to release resource after alarm related data is obtained.

#### Close the search handle to alarm related data

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseAlarmRelatedDataList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching alarm related data and release resource.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_FindAlarmRelatedDataList](#_获取指定告警的关联数据列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindAlarmRelatedDataList](#_获取指定告警的关联数据列表) and [NETDEV\_FindNextAlarmRelatedDataInfo](#_获取指定告警的关联数据) interfaces.
* Call the [NETDEV\_FindCloseAlarmRelatedDataList](#_关闭查询指定告警的关联数据) interface to release resource after alarm related data is obtained.

### Register the callback function to receive alarm images

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPicAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_PicAlarmMessCallBack\_PF cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive alarm images

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbAlarmMessCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_PicAlarmMessCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_ALARM\_PIC\_DATA\_S](#_告警图片数据) pstAlarmPicData,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstAlarmPicData | IN | Alarm image info |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### People counting

#### Register the callback function to receive people counting alarms

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPeopleCountAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_PeopleCountAlarmMessCallBack\_PF cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive people counting alarms

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbAlarmMessCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_PeopleCountAlarmMessCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_PEOPLE\_COUNT\_ALARM\_INFO\_S](#_人数统计告警信息) pstAlarmInfo,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstAlarmInfo | IN | Alarm image info |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Get crowd density group list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindCrowdDensityGroupList  (  LPVOID lpUserID  ); |

**Interface description:**

Get crowd density group list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |

**Return value:**

Null means failure, other values are used as the input parameter of [NETDEV\_FindNextCrowdDensityGroupInfo](#_逐个获取查找到的人员密度报警统计组信息) and [NETDEV\_FindCloseCrowdDensityGroupList](#_逐个获取查找到的人员密度报警统计组信息). Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextCrowdDensityGroupInfo](#_逐个获取查找到的人员密度报警统计组信息) and [NETDEV\_FindCloseCrowdDensityGroupList](#_逐个获取查找到的人员密度报警统计组信息) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextCrowdDensityGroupInfo](#_逐个获取查找到的人员密度报警统计组信息) interface repeatedly to get the next group info.
* Call the [NETDEV\_FindCloseCrowdDensityGroupList](#_逐个获取查找到的人员密度报警统计组信息) interface to release resource after crowd density group info is obtained.

#### Get crowd density group info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextCrowdDensityGroupInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_CROWD\_DENSITY\_GROUP\_INFO\_S](#_人员密度报警统计组信息) pstCrowdDensityGroupInfo  ); |

**Interface description:**

Get object search result

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_FindCrowdDensityGroupList](#_查询人员密度报警统计组列表信息) interface. |
| pstCrowdDensityGroupInfo | IN | Crowd density group info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindCrowdDensityGroupList](#_查询人员密度报警统计组列表信息) and [NETDEV\_FindCloseCrowdDensityGroupList](#_逐个获取查找到的人员密度报警统计组信息) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseCrowdDensityGroupList](#_逐个获取查找到的人员密度报警统计组信息) interface to release resource after crowd density group info is obtained.

#### Close the search handle to crowd density group list

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseCrowdDensityGroupList  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching crowd density group and release resource

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Search handle, which is the return value from the [NETDEV\_FindCrowdDensityGroupList](#_查询人员密度报警统计组列表信息) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindCrowdDensityGroupList](#_查询人员密度报警统计组列表信息) and [NETDEV\_FindNextCrowdDensityGroupInfo](#_逐个获取查找到的人员密度报警统计组信息) interfaces.
* Call the [NETDEV\_FindCloseCrowdDensityGroupList](#_逐个获取查找到的人员密度报警统计组信息) interface to release resource after crowd density group info is obtained.

#### Add crowd density group

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddCrowdDensityGroupInfo  (  LPVOID lpUserID,  [LPNETDEV\_CROWD\_DENSITY\_GROUP\_INFO\_S](#_人员密度报警统计组信息) pstCrowdDensityGroupInfo,  UINT32 \*pUdwGroupID  ); |

**Interface description:**

Add crowd density group

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstCrowdDensityGroupInfo | IN | Crowd density group info |
| pUdwGroupID | OUT | Crowd density group ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Modify crowd density group

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyCrowdDensityGroupInfo  (  LPVOID lpUserID,  [LPNETDEV\_CROWD\_DENSITY\_GROUP\_INFO\_S](#_人员密度报警统计组信息) pstCrowdDensityGroupInfo  ); |

**Interface description:**

Modify crowd density group

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstCrowdDensityGroupInfo | IN | Crowd density group info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Delete crowd density group

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteCrowdDensityGroupInfo  (  LPVOID lpUserID,  UINT32 udwCrowdDensityGroupID  ); |

**Interface description:**

Delete crowd density group

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwCrowdDensityGroupID | IN | Crowd density group ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Get info about crowd density group

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetCrowdDensityGroupInfo  (  LPVOID lpUserID,  UINT32 udwCrowdDensityGroupID,  [LPNETDEV\_CROWD\_DENSITY\_GROUP\_INFO\_S](#_人员密度报警统计组信息) pstCrowdDensityGroupInfo  ); |

**Interface description:**

Get info about crowd density group

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwCrowdDensityGroupID | IN | Crowd density group ID |
| pstCrowdDensityGroupInfo | OUT | Crowd density group info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Reset tripwire people counting statistics immediately

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_ResetLinesPeopleCounting  (  LPVOID lpUserID,  UINT32 udwChannelID  ); |

**Interface description:**

Reset tripwire people counting statistics immediately

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| udwChannelID | IN | Channel ID |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Thermal Imaging

#### Register the callback function to receive fire detection alarms

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetConflagrationAlarmCallBackV30  (  LPVOID lpUserID,  NETDEV\_ConflagrationAlarmCallBack\_PF\_V30 cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive fire detection alarms

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbAlarmMessCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \* NETDEV\_ConflagrationAlarmCallBack\_PF\_V30)  (  LPVOID lpUserID,  [LPNETDEV\_CONFLAGRATION\_ALARM\_INFO\_V30\_S](#_火点告警信息结构体) pstAlarmInfo,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstAlarmInfo | IN | Fire detection alarm info |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Register the callback function to receive smoking detection alarms

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetSmokeAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_SmokeAlarmMessCallBack\_PF cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive smoking detection alarms.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbAlarmMessCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_SmokeAlarmMessCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_SMOKE\_DETC\_S](#_吸烟检测信息结构体) pstSmokeAlarmData,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstSmokeAlarmData | IN | Smoking detection alarm info. |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

#### Register the callback function to receive temperature measurement alarms

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetTemperatureDetcCallBack  (  LPVOID lpUserID,  NETDEV\_TemperatureDetcMessCallBack\_PF cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**Interface description:**

Register the callback function to receive temperature alarms

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| cbAlarmMessCallBack | IN | Callback function |
| lpUserData | IN | User data |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_TemperatureDetcMessCallBack\_PF)  (  LPVOID lpUserID,  [LPNETDEV\_TEMPERATURE\_ALARM\_INFO\_S](#_温度检测告警信息) pstTemperatureDetcInfo,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstTemperatureDetcInfo | IN | Temperature detection alarm info |
| lpUserData | IN | User data |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

## System Maintenance

### Search logs

#### Search logs

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindLogInfoList  (  LPVOID lpUserID,  [LPNETDEV\_FIND\_LOG\_COND\_S](#_日志查询条件结构体) pstLogFindCond  ); |

**Interface description:**

Search logs by log type and time

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstLogFindCond | IN | Log search criteria |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextLogInfo](#_逐个查找日志信息) and [NETDEV\_FindCloseLogInfo](#_关闭日志列表查找).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextLogInfo](#_逐个查找日志信息) and [NETDEV\_FindCloseLogInfo](#_关闭日志列表查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextLogInfo](#_逐个查找日志信息) interface repeatedly to get the next log.
* Call the [NETDEV\_FindCloseLogInfo](#_关闭日志列表查找) interface to release resource and stop the search after log info is obtained.
* This interface is only applicable to VMS and NVR. VMS only supports searching operation logs.

**See also:**

[NETDEV\_FindNextLogInfo](#_逐个查找日志信息), [NETDEV\_FindCloseLogInfo](#_关闭日志列表查找)

#### Get log info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextLogInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_LOG\_INFO\_S](#_日志信息结构体) pstLogInfo  ); |

**Interface description:**

Get log info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Log search handle, which is the return value from the [NETDEV\_FindLogInfoList](#_查找日志列表) interface. |
| pstLogInfo | OUT | Pointer to the saved log info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindLogInfoList](#_查找日志列表) and [NETDEV\_FindCloseLogInfo](#_关闭日志列表查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseLogInfo](#_关闭日志列表查找) interface to release resource and stop the search after log info is obtained.
* This interface is only applicable to VMS and NVR. VMS only supports searching operation logs.

**See also:**

[NETDEV\_FindLogInfoList](#_查找日志列表), [NETDEV\_FindCloseLogInfo](#_关闭日志列表查找)

#### Stop searching logs and release resource

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseLogInfo  (  LPVOID lpFindHandle  );s |

**Interface description:**

Stop searching log and release resource.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Log search handle, which is the return value from the [NETDEV\_FindLogInfoList](#_查找日志列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindLogInfoList](#_查找日志列表) and [NETDEV\_FindNextLogInfo](#_逐个查找日志信息) interfaces.
* Call the [NETDEV\_FindCloseLogInfo](#_关闭日志列表查找) interface to release resource and stop the search after log info is obtained.

**See also:**

[NETDEV\_FindLogInfoList](#_查找日志列表), [[NETDEV\_FindNextLogInfo](#_逐个查找日志信息)o](#_关闭日志列表查找)

### Get device alarms

#### Get device alarm list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindAlarmInfoList  (  LPVOID lpUserID,  [LPNETDEV\_FIND\_ALARM\_COND\_S](#_告警信息查找条件结构体) pstAlarmFindCond  ); |

**Interface description:**

Get device alarms by time

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstAlarmFindCond | IN | Alarm search criteria |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息) and [NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息) and [NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息) interface repeatedly to get the next alarm info.
* Call the [NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找) interface to release resource and stop the search after alarm info is obtained.

**See also:**

[NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息), [NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找)

#### Get alarm info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextAlarmInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_FIND\_ALARM\_INFO\_S](#_告警信息结构体) pstAlarmInfo  ); |

**Interface description:**

Get alarm info one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Alarm search handle, which is the return value from the [NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表) interface. |
| pstAlarmInfo | OUT | Alarm info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表) and [NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找) interface to release resource and stop the search after alarm info is obtained.

**See also:**

[NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表), [NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找)

#### Stop searching alarm info

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseAlarmInfo  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching alarm and release resource.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Alarm search handle, which is the return value from the [NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表) and [NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息) interfaces.
* Call the [NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找) interface to release resource and stop the search after alarm info is obtained.

**See also:**

[NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表), [NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息)

### Get alarm logs

#### Get alarm log list

**Interface name:**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindAlarmLogList  (  LPVOID lpUserID,  [LPNETDEV\_ALARM\_LOG\_COND\_LIST\_S](#_告警日志查询条件列表) pstFindCond,  INT32\* pdwTotalRealRow  ); |

**Interface description:**

Get alarm log list

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |
| pstFindCond | IN | Search criteria |
| pdwTotalRealRow | OUT | Actual total number of alarm logs |

**Return value:**

Null means failure, other values are used as input parameters of [NETDEV\_FindNextAlarmLog](#_逐个获取告警日志信息) and [NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找).  
Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindNextAlarmLog](#_逐个获取告警日志信息) and [NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找) interfaces.
* After the interface is called successfully, call the [NETDEV\_FindNextAlarmLog](#_逐个获取告警日志信息) interface repeatedly to get the next alarm log.
* Call the [NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找) interface to release resource and stop the search after alarm log is obtained.
* This interface is only applicable to VMS.

**See also:**

[NETDEV\_FindNextAlarmLog](#_逐个获取告警日志信息), [NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找)

#### Get alarm log info one by one

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextAlarmLog  (  LPVOID lpFindHandle,  LPNETDEV\_ALARM\_LOG\_INFO\_S pstAlarmLogInfo  ); |

**Interface description:**

Get alarm log one by one

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Alarm search handle, which is the return value from the [NETDEV\_FindAlarmLogList](#_获取告警日志列表) interface. |
| pstAlarmLogInfo | OUT | Alarm log info |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindAlarmLogList](#_获取告警日志列表) and [NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找) interfaces.
* Call this interface repeatedly to get the next info.
* Call the [NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找) interface to release resource and stop the search after alarm log is obtained.

**See also:**

[NETDEV\_FindAlarmLogList](#_获取告警日志列表), [NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找)

#### Stop searching alarm logs

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseAlarmLog  (  LPVOID lpFindHandle  ); |

**Interface description:**

Stop searching alarm log and release resource.

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpFindHandle | IN | Alarm search handle, which is the return value from the [NETDEV\_FindAlarmLogList](#_获取告警日志列表) interface. |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Use this interface with the [NETDEV\_FindAlarmLogList](#_获取告警日志列表) and [NETDEV\_FindNextAlarmLog](#_逐个获取告警日志信息) interfaces.
* Call the [NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找) interface to release resource and stop the search after alarm log is obtained.

### Restart device

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_Reboot  (  PVOID lpUserID  ); |

**Interface description:**

Restart device

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

### Restore device to factory defaults

**Interface name:**

|  |
| --- |
| BOOL STDCALL NETDEV\_RestoreConfig  (  LPVOID lpUserID  ); |

**Interface description:**

Restore factory defaults

**Parameters:**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpUserID | IN | User login handle, which is the return value from the [NETDEV\_Login\_V30](#_用户设备登录) interface |

**Return value:**

TRUE means success, other values mean failure. Call [NETDEV\_GetLastError](#_获取接口错误码_1) to get the error code and use the error code to determine the cause of error.

Remarks:

* Keep current network and user configurations, and restore others to factory defaults.

## Structure Definition

### Structure of device login info

|  |
| --- |
| typedef struct tagstNETDEVDeviceLoginInfo  {  CHAR          szIPAddr[NETDEV\_LEN\_260];  INT32           dwPort;  CHAR          szUserName[NETDEV\_LEN\_132];  CHAR          szPassword[NETDEV\_LEN\_128];  INT32           dwLoginProto;  INT32           dwDeviceType;  BYTE           byRes[256];  }NETDEV\_DEVICE\_LOGIN\_INFO\_S, \*LPNETDEV\_DEVICE\_LOGIN\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szIPAddr | Device IP address/domain name |
| dwPort | Port number, the same port number used for web access; the default is 80. |
| szUserName | Username |
| szPassword | Password. The user enters the plain password, which is encrypted inside the SDK. |
| dwLoginProto | Login protocol. See [NETDEV\_LOGIN\_PROTO\_E](#_接入协议枚举). |
| dwDeviceType | Device type. See [NETDEV\_DEVICE\_TYPE\_E](#_错误码列表). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_Login\_V30](#_用户设备登录)

### Structure of security login info

|  |
| --- |
| typedef struct tagstNETDEVSELogInfo  {  INT32    dwSELogCount;  INT32    dwSELogTime;  BYTE   byRes[64];  }NETDEV\_SELOG\_INFO\_S, \*LPNETDEV\_SELOG\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSELogCount | Secure login count. For example, how many login attempts are allowed (incorrect username/password) before the device will be locked. |
| dwSELogTime | Secure login time. For example, how much time after the device is locked can the user retry login. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_Login\_V30](#_用户设备登录)

### Structure of timeout

|  |
| --- |
| typedef struct tagNETDEVRevTimeout  {  INT32 dwRevTimeOut;  INT32 dwFileReportTimeOut;  BYTE byRes[128];  }NETDEV\_REV\_TIMEOUT\_S, \*LPNETDEV\_REV\_TIMEOUT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwRevTimeOut | Network message timeout value (unit: second) |
| dwFileReportTimeOut | File transfer timeout value (unit: second) |
| byRes | Reserved field. |

**Remarks:**

* The dwRevTimeOut must be greater than or equal to 1 second.
* The dwFileReportTimeOut must be greater than or equal to 30 seconds.

**See also:**

[NETDEV\_SetRevTimeOut](#_设置超时时间)

### Structure of discovered device info

|  |
| --- |
| typedef struct tagNetDEVDiscoveryDevInfo  {  CHAR szDevAddr[NETDEV\_LEN\_64];  CHAR szDevModule[NETDEV\_LEN\_64];  CHAR szDevSerailNum[NETDEV\_LEN\_64];  CHAR szDevMac[NETDEV\_LEN\_64];  CHAR szDevName[NETDEV\_LEN\_64];  CHAR szDevVersion[NETDEV\_LEN\_64];  [NETDEV\_DEVICE\_TYPE\_E](#_错误码列表) enDevType;  INT32 dwDevPort;  CHAR szManuFacturer[NETDEV\_LEN\_64];  CHAR szActiveCode[NETDEV\_LEN\_64];  CHAR szCloudUserName[NETDEV\_LEN\_64];  BYTE byRes[68];  }NETDEV\_DISCOVERY\_DEVINFO\_S, \*LPNETDEV\_DISCOVERY\_DEVINFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szDevAddr | Device address |
| szDevModule | Device model |
| szDevSerailNum | Device serial number |
| szDevMac | Device MAC address |
| szDevName | Device name |
| szDevVersion | Device version |
| enDevType | Device type. See [NETDEV\_DEVICE\_TYPE\_E](#_错误码列表). |
| dwDevPort | Device port number |
| szManuFacturer | Manufacturer |
| szActiveCode | Activation code |
| szCloudUserName | Cloud username |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SetDiscoveryCallBack](#_设置设备搜索回调函数)

### Structure of abnormal player output info

|  |
| --- |
| typedef struct tagstNETDEVExceptionOutputInfo  {  INT32 dwEventCode;  INT64 tPlayBackTime;  CHAR szFileName[NETDEV\_LEN\_256];  INT32 dwOldSubID;  INT32 dwNewSubID;  INT32 dwSubType;  BYTE byRes[116];  }NETDEV\_EXCEPTION\_OUTPUT\_INFO\_S, \*LPNETDEV\_EXCEPTION\_OUTPUT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwEventCode | Event code (for player to report events) |
| tPlayBackTime | Playback absolute time (for player to report events) |
| szFileName | String parameter |
| dwOldSubID | Old subscription ID, only used for face subscription |
| dwNewSubID | New subscription ID, only used for face subscription |
| dwSubType | Subscription type, only used for face subscription |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SetExceptionCallBack](#_注册异常回调函数)

### Structure of device address info

|  |
| --- |
| typedef struct tagNETDEVDevAddrInfo  {  CHAR szUserName[NETDEV\_LEN\_64];  CHAR szPassword[NETDEV\_LEN\_64];  CHAR szIPv4Address[NETDEV\_LEN\_32];  CHAR szIPv4GateWay[NETDEV\_LEN\_32];  CHAR szIPv4SubnetMask[NETDEV\_LEN\_32];  CHAR szDevSerailNum[NETDEV\_LEN\_64];  CHAR szDevMac[NETDEV\_LEN\_64];  BYTE byRes[512];  }NETDEV\_DEV\_ADDR\_INFO\_S, \*LPNETDEV\_DEV\_ADDR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szUserName | Username |
| szPassword | Password |
| szIPv4Address | IP address (IPv4) |
| szIPv4GateWay | Gateway address (IPv4) |
| szIPv4SubnetMask | Subnet mask (IPv4 ) |
| szDevSerailNum | Device serial number |
| szDevMac | Device MAC address |
| byRes | Reserved field. |

**See also:**

[NETDEV\_ModifyDeviceAddr](#_修改设备IP地址)

### Structure of live view parameters

|  |
| --- |
| typedef struct tagNETDEVPriviewInfo  {  INT32 dwChannelID;  INT32 dwStreamType;  INT32 dwLinkMode;  LPVOID hPlayWnd;  INT32 dwFluency;  INT32 dwStreamMode;  INT32 dwLiveMode;  INT32 dwDisTributeCloud;  BOOL dwallowDistribution;  INT32 dwTransType;  INT32 dwStreamProtocol;  BYTE byRes[236];  }NETDEV\_PREVIEWINFO\_S, \*LPNETDEV\_PREVIEWINFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| dwStreamType | Stream type. See [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举). |
| dwLinkMode | Transport protocol. See [NETDEV\_PROTOCAL\_E](#_媒体传输协议枚举). |
| hPlayWnd | Window handle |
| dwFluency | Fluency priority type. See [NETDEV\_PICTURE\_FLUENCY\_E](#_图像播放流畅性枚举). |
| dwStreamMode | Streaming mode. See [NETDEV\_STREAM\_MODE\_E](#_起流模式枚举). |
| dwLiveMode | Streaming mode. Reserved field. Just pass in 0. |
| dwDisTributeCloud | Distribution capability. Reserved field. Just pass in 0. |
| dwallowDistribution | Whether CDN is supported. Reserved field. Pass 0. |
| dwTransType | Transport type. Applicable to VMS only. See [NETDEV\_TRANS\_TYPE\_E](#_传输类型枚举). |
| dwStreamProtocol | Streaming protocol. See [NETDEV\_START\_STREAM\_PROT\_E](#_起流协议枚举). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_RealPlay](#_实时预览)

### Structure of decoded audio data

|  |
| --- |
| ttypedef struct tagNETDEVWaveData  {  CHAR \*pcData;  INT32 dwDataLen;  INT32 dwWaveFormat;  }NETDEV\_WAVE\_DATA\_S, \*LPNETDEV\_WAVE\_DATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| pcData | Audio data |
| dwDataLen | Length of audio data |
| dwWaveFormat | Decoded audio format. See [NETDEV\_WAVE\_FORMAT\_INFO\_E](#_解码后音频格式). |

**See also:**

[NETDEV\_StartVoiceCom](#_开启语音对讲)

### Structure of audio parameters

|  |
| --- |
| typedef struct tagPlayerAudioSampleParamType  {  ULONG\_32 ulChannels;  ULONG\_32 ulSampleRate;  [PLAYER\_AUDIO\_SAMPLE\_FORMAT\_E](#_音频位宽枚举) enSampleFormat;  } PLAYER\_AUDIO\_SAMPLE\_PARAM\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| ulChannels | Number of audio channels, 1: mono, 2: stereo |
| ulSampleRate | Sampling rate |
| enSampleFormat | Bit width |

**See also:**

[NETDEV\_InputVoiceData](#_语音数据转发)

### Structure of channel operation info

|  |
| --- |
| typedef struct tagNETDEVOperateInfo  {  INT32 dwID;  INT32 dwReturnCode;  BYTE byRes[64];  }NETDEV\_OPERATE\_INFO\_S, \*LPNETDEV\_OPERATE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwID | Action ID |
| dwReturnCode | Return code |
| byRes | Reserved field. |

### Structure of channel list

|  |
| --- |
| typedef struct tagNETDEVOperateList  {  INT32 dwSize;  [LPNETDEV\_OPERATE\_INFO\_S](#_单个操作信息) pstOperateInfo;  }NETDEV\_OPERATE\_LIST\_S, \*LPNETDEV\_OPERATE\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Batch processing size |
| pstOperateInfo | Batch processing info |

**See also:**

[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)

### Structure of info about batch processing audio broadcast group

|  |
| --- |
| typedef struct NETDEV\_ModifyVoiceBroadcastInfo  {  [LPNETDEV\_OPERATE\_LIST\_S](#_批处理列表) pstAddChannelList;  [LPNETDEV\_OPERATE\_LIST\_S](#_批处理列表) pstDelChannelList;  }NETDEV\_MODIFY\_VOICE\_BROADCAST\_INFO\_S, \*LPNETDEV\_MODIFY\_VOICE\_BROADCAST\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| pstAddChannelList | Add channel list |
| pstDelChannelList | Delete channel list |

**See also:**

[NETDEV\_ModifyVoiceBroadcastGroup](#_修改语音广播组)

### Structure of info about audio broadcast channels

|  |
| --- |
| typedef struct tagstNETDEV\_VoiceBroadcastChlInfo  {  UINT32 dwChannelID;  UINT32 dwResultCode;  UINT32 dwStatus;  }NETDEV\_VOICE\_BROADCAST\_CHL\_INFO\_S, \*LPNETDEV\_VOICE\_BROADCAST\_CHL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Audio channel ID |
| dwResultCode | Broadcast channel status code |
| dwStatus | Audio status of broadcast channel |

### Structure of info about audio broadcast group channels

|  |
| --- |
| typedef struct tagstNETDEV\_VoiceBroadcastGroupInfo  {  INT32 dwSize;  [NETDEV\_VOICE\_BROADCAST\_CHL\_INFO\_S](#_语音广播通道信息结构体) astChlInfoList[NETDEV\_VOICE\_BROADCAST\_CHANNEL\_NUM\_MAX];  }NETDEV\_VOICE\_BROADCAST\_GROUP\_INFO\_S, \*LPNETDEV\_VOICE\_BROADCAST\_GROUP\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Number of broadcast group channels |
| astChlInfoList | All channel info |

**See also:**

[NETDEV\_GetVoiceBroadcastChlStatus](#_获取语音广播组通道状态)

### Structure of info about audio broadcast group control

|  |
| --- |
| typedef struct tagstNETDEV\_VoiceBroadcastCtrlInfo  {  INT32 dwChannelID;  UINT32 udwStatus;  }NETDEV\_VOICE\_BROADCAST\_CTRL\_INFO\_S, \*LPNETDEV\_VOICE\_BROADCAST\_CTRL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| udwStatus | Audio status. 1: On, 0: Off |

**See also:**

[NETDEV\_ModifyVoiceBroadcastStatus](#_修改语音广播组通道状态)

### Structure of audio data

|  |
| --- |
| typedef struct tagNETDEVWaveData  {  CHAR \*pcData;  INT32 dwDataLen;  INT32 dwWaveFormat;  }NETDEV\_WAVE\_DATA\_S, \*LPNETDEV\_WAVE\_DATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| pcData | Audio data |
| dwDataLen | Length of audio data |
| dwWaveFormat | Decoded audio format. See [NETDEV\_WAVE\_FORMAT\_INFO\_E](#_解码后音频格式枚举). |

**See also:**

[NETDEV\_SetPlayDecodeAudioCB](#_注册解码后音频数据回调)

### Structure of image data

|  |
| --- |
| typedef struct tagNETDEVPictureData  {  BYTE \*pucData[4];  INT32 dwLineSize[4];  INT32 dwPicHeight;  INT32 dwPicWidth;  INT32 dwRenderTimeType;  INT64 tRenderTime;  }NETDEV\_PICTURE\_DATA\_S, \*LPNETDEV\_PICTURE\_DATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| pucData | YUV data, pucData[0]: Y-plane pointer, pucData[1]: U-plane pointer, pucData[2]: V-plane pointer |
| dwLineSize | dwLineSize [0]: Y-plane line size, dwLineSize [1]: U-plane line size, dwLineSize [2]: V-plane line size |
| dwPicHeight | Image height |
| dwPicWidth | Image width |
| dwRenderTimeType | Time data type used for rendering. |
| tRenderTime | Time data used for rendering. |

**See also:**

[NETDEV\_SetPlayDecodeVideoCB](#_注册解码后视频数据回调)

### Structure of assembled audio data

|  |
| --- |
| typedef struct tagNETDEVParseAudioData  {  BYTE \*pucData;  UINT32 udwDataLen;  UINT32 udwAudioCodeFormat;  UINT32 udwWaveFormat;  INT64 tTimeStamp;  BYTE byRes[8];  }NETDEV\_PARSE\_AUDIO\_DATA\_S, \*LPNETDEV\_PARSE\_AUDIO\_DATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| pucData | Audio data |
| udwDataLen | Length of audio data |
| udwAudioCodeFormat | Audio code format. See [NETDEV\_AUDIO\_ENCODE\_TYPE\_E](#_音频编码格式枚举). |
| udwWaveFormat | Decoded audio format. See [NETDEV\_WAVE\_FORMAT\_INFO\_E](#_解码后音频格式). |
| tTimeStamp | Timestamp |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SetPlayParseAudioCB](#_注册拼帧后音频数据回调)

### Structure of parsed video data

|  |
| --- |
| typedef struct tagNETDEVParseVideoData  {  BYTE \*pucData;  INT32 dwDataLen;  INT32 dwVideoFrameType;  INT32 dwVideoCodeFormat;  INT32 dwHeight;  INT32 dwWidth;  INT64 tTimeStamp;  INT64 tAbTime;  }NETDEV\_PARSE\_VIDEO\_DATA\_S, \*LPNETDEV\_PARSE\_VIDEO\_DATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| pucData | Video data |
| dwDataLen | Video data length |
| dwVideoFrameType | Video frame type. See [NETDEV\_VIDEO\_FRAME\_TYPE\_E](#_视频帧类型枚举). |
| dwVideoCodeFormat | Video encoding format. See [NETDEV\_VIDEO\_CODE\_TYPE\_E](#_视频编码格式枚举). |
| dwHeight | Video image height |
| dwWidth | Video image width |
| tTimeStamp | Timestamp (ms) |
| tAbTime | Absolute time (unix timestamp). Currently only exists in playback streams. |

**See also:**

[NETDEV\_SetPlayParseCB](#_注册拼帧后视频数据回调)

### Structure of image info

|  |
| --- |
| typedef struct tagNETDEVVideoEffect  {  INT32 dwContrast;  INT32 dwBrightness;  INT32 dwSaturation;  INT32 dwHue;  INT32 dwGamma;  BYTE byRes[16];  }NETDEV\_VIDEO\_EFFECT\_S, \*LPNETDEV\_VIDEO\_EFFECT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwContrast | Contrast |
| dwBrightness | Brightness |
| dwSaturation | Saturation |
| dwHue | Hue |
| dwGamma | Gamma |
| byRes | Reserved field. |

**See also:**

[NETDEV\_GetVideoEffect](#_获取影像参数), [NETDEV\_SetVideoEffect](#_影像调节)

### Structure of rectangular area

|  |
| --- |
| typedef struct tagNETDEVRect  {  INT32 dwLeft;  INT32 dwTop;  INT32 dwRight;  INT32 dwBottom;  }NETDEV\_RECT\_S, \*LPNETDEV\_RECT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwLeft | x-coordinate of top left [0,10000] |
| dwTop | y-coordinate of top left [0,10000] |
| dwRight | x-coordinate of bottom right [0,10000] |
| dwBottom | y-coordinate of bottom right [0,10000] |

**See also:**

[NETDEV\_SetDigitalZoom](#_设置数字放大)

### Structure of recording search info

|  |
| --- |
| typedef struct tagNETDEVFindCond  {  CHAR szFileName[NETDEV\_LEN\_64];  INT32 dwChannelID;  INT32 dwStreamType;  INT32 dwFileType;  INT64 tBeginTime;  INT64 tEndTime;  INT32 dwRecordLocation;  UINT32 udwServerID;  BYTE byRes[28];  }NETDEV\_FILECOND\_S, \*LPNETDEV\_FILECOND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szFileName | Recording file name |
| dwChannelID | Channel ID |
| dwStreamType | Stream type. See [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举). |
| dwFileType | Recording storage type. See [NETDEV\_STORE\_TYPE\_E](#_录像存储类型枚举). |
| tBeginTime | Start time |
| tEndTime | End time |
| dwRecordLocation | Recording storage location. See [NETDEV\_RECORD\_LOCATION\_E](#_录像存储位置枚举). |
| udwServerID | Recording server ID. Applicable to VMS only. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_FindFile](#_查找录像文件列表)

### Structure of recording file info

|  |
| --- |
| typedef struct tagNETDEVFindData  {  CHAR szFileName[NETDEV\_FILE\_NAME\_LEN];  INT64 tBeginTime;  INT64 tEndTime;  BYTE byFileType;  UINT32 udwServerID;  UINT32 udwFileSize;  INT32 dwFileType;  BYTE byRes[159  }NETDEV\_FINDDATA\_S,\*LPNETDEV\_FINDDATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szFileName | Recording file name |
| tBeginTime | Start time |
| tEndTime | End time |
| byFileType | Recording storage type. See [NETDEV\_STORE\_TYPE\_E](#_录像存储类型枚举). |
| udwServerID | Recording server ID. Applicable to VMS only. |
| udwFileSize | Recording file size |
| dwFileType | File type. See [NETDEV\_RECORD\_SEARCH\_TYPE\_E](#_按位查询录像类型枚举). Currently for internal use only. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_FindNextFile](#_逐个获取查找到的文件信息)

### Structure of info about searching recording location by month

|  |
| --- |
| typedef struct tagNETDEVMonthInfo  {  UINT32 udwYear;  UINT32 udwMonth;  UINT32 udwPosition;  }NETDEV\_MONTH\_INFO\_S, \*LPNETDEV\_MONTH\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwYear | Year |
| udwMonth | Month |
| udwPosition | Recording search location. See [NETDEV\_VIDEO\_POSITION\_E](#_录像查询位置枚举). |

**See also:**

[NETDEV\_QuickSearch](#_单通道查询), [NETDEV\_QuickSearchMultiChl](#_多通道查询)

### Structure of recording status by month

|  |
| --- |
| typedef struct tagNETDEVMonthStatus  {  UINT32 udwDayNumInMonth;  UINT32 szVideoStatus[NETDEV\_MONTH\_DAY\_MAX];  }NETDEV\_MONTH\_STATUS\_S, \*LPNETDEV\_MONTH\_STATUS\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwDayNumInMonth | Actual number of days in the month |
| szVideoStatus | Recording status list. See [NETDEV\_VIDEO\_STATUS\_E](#_录像状态枚举). |

**See also:**

[NETDEV\_QuickSearch](#_单通道查询), [NETDEV\_QuickSearchMultiChl](#_多通道查询)

### Structure of channel list

|  |
| --- |
| typedef struct tagNETDEVChannels  {  UINT32 udwSize;  INT32 dwChannelIDs[NETDEV\_CHANNEL\_MAX];  }NETDEV\_CHANNELS\_S, \*LPNETDEV\_CHANNELS\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSize | Number of channels |
| dwChannelIDs | Video input channel ID list to be searched |

**See also:**

[NETDEV\_QuickSearch](#_单通道查询), [NETDEV\_QuickSearchMultiChl](#_多通道查询)

### Structure of playback by time parameters

|  |
| --- |
| typedef struct tagNETDEVPlayBackCondition  {  INT32 dwChannelID;  INT64 tBeginTime;  INT64 tEndTime;  INT32 dwLinkMode;  LPVOID hPlayWnd;  INT32 dwFileType;  INT32 dwDownloadSpeed;  INT32 dwStreamMode;  INT32 dwStreamIndex;  INT32 dwRecordLocation;  INT32 dwTransType;  BOOL bCloudStorage;  BOOL bOneFrameEnable;  INT32 dwPlaySpeed;  NETDEV\_DECODE\_VIDEO\_DATA\_CALLBACK\_PF cbPlayDecodeVideoCALLBACK;  INT64 tPlayTime;  UINT32 udwServerID;  BYTE byRes[212];  }NETDEV\_PLAYBACKCOND\_S, \*LPNETDEV\_PLAYBACKCOND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| tBeginTime | Start time |
| tEndTime | End time |
| dwLinkMode | Transport protocol. See [NETDEV\_PROTOCAL\_E](#_媒体传输协议枚举). |
| hPlayWnd | Window handle |
| dwFileType | Recording storage type. See [NETDEV\_STORE\_TYPE\_E](#_录像存储类型枚举). |
| dwDownloadSpeed | Download speed. Valid when downloading recordings. See [NETDEV\_E\_DOWNLOAD\_SPEED\_E](#_录像下载速度枚举). |
| dwStreamMode | Streaming mode. See [NETDEV\_STREAM\_MODE\_E](#_起流模式枚举). |
| dwStreamIndex | Stream type. See [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举). |
| dwRecordLocation | Recording storage location. Applicable to VMS only. See [NETDEV\_RECORD\_LOCATION\_E](#_录像存储位置枚举). |
| dwTransType | Transport type. Applicable to VMS only. See [NETDEV\_TRANS\_TYPE\_E](#_传输类型枚举). |
| bCloudStorage | Enable or disable cloud storage based playback. Reserved parameter. For internal use only. |
| bOneFrameEnable | Enable or disable frame-by-frame decoding mode. Decoding efficiency will be affected when enabled |
| dwPlaySpeed | Playback speed. See [NETDEV\_VOD\_PLAY\_STATUS\_E](#_播放/下载速度枚举). |
| cbPlayDecodeVideoCALLBACK | Decoded data callback function |
| tPlayTime | Play time |
| udwServerID | Recording server ID. Applicable to VMS only. |
| byRes | Reserved field. |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL\* NETDEV\_DECODE\_VIDEO\_DATA\_CALLBACK\_PF)  (  LPVOID lpPlayHandle,  const [NETDEV\_PICTURE\_DATA\_S](#_媒体数据流格式) \*pstPictureData,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| lpPlayHandle | IN | Current live view/playback handle |
| pstPictureData | IN | Buffer storing info about decoded video stream data. See [NETDEV\_PICTURE\_DATA\_S](#_媒体数据流格式). |
| lpUserParam | IN | User data. User parameter specified when calling calling NETDEV\_SetPlayDecodeVideoCB. |

**See also:**

[NETDEV\_FastPlayBackByUrl](#_根据URL回放录像文件), [NETDEV\_PlayBackByTime](#_按时间回放录像文件), [NETDEV\_GetFileByTime](#_按时间下载录像文件)

### Structure of playback by filename parameters

|  |
| --- |
| typedef struct tagNETDEVPlayBackInfo  {  char szName[NETDEV\_LEN\_260];  INT64 tBeginTime;  INT64 tEndTime;  INT32 dwLinkMode;  LPVOID hPlayWnd;  INT32 dwFileType;  INT32 dwDownloadSpeed;  INT32 dwStreamMode;  BYTE byRes[252];  }NETDEV\_PLAYBACKINFO\_S, \*LPNETDEV\_PLAYBACKINFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szName | Playback control block name |
| tBeginTime | Start time |
| tEndTime | End time |
| dwLinkMode | Transport protocol. See [NETDEV\_PROTOCAL\_E](#_媒体传输协议枚举). |
| hPlayWnd | Window handle |
| dwFileType | Recording storage type. Reserved and currently not in use. Keep the default 0. |
| dwDownloadSpeed | Download speed. Valid when downloading recordings. See [NETDEV\_E\_DOWNLOAD\_SPEED\_E](#_录像下载速度枚举). |
| dwStreamMode | Streaming mode. See [NETDEV\_STREAM\_MODE\_E](#_起流模式枚举). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_PlayBackByTime](#_按时间回放录像文件), [NETDEV\_GetFileByName](#_按文件名下载录像文件)

### Structure of drag-to-zoom

|  |
| --- |
| typedef struct tagNETDEVPTZZoomAreaInfo  {  UINT32 udwMidPointX;  UINT32 udwMidPointY;  UINT32 udwLengthX;  UINT32 udwLengthY;  UINT32 udwWidth;  UINT32 udwHeight;  BYTE byRes[64];  }NETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S, \*LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwMidPointX | x-coordinate of the center of the zoomed area |
| udwMidPointY | y-coordinate of the center of the zoomed area |
| udwLengthX | Length of the zoomed area |
| udwLengthY | Width of the zoomed area |
| udwWidth | Window length |
| udwHeight | Window height |
| byRes | Reserved field. |

**See also:**

[NETDEV\_PTZ3DPosition](#_云台3D定位_1)

### Structure of info about PTZ areas

|  |
| --- |
| typedef struct tagNETDEVPtzOperateAreaInfo  {  INT32 dwBeginPointX;  INT32 dwBeginPointY;  INT32 dwEndPointX;  INT32 dwEndPointY;  }NETDEV\_PTZ\_OPERATEAREA\_S, \*LPNETDEV\_PTZ\_OPERATEAREA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwBeginPointX | x-coordinate of the start point of the area [0,10000] |
| dwBeginPointY | y-coordinate of the start point of the area [0,10000] |
| dwEndPointX | x-coordinate of the end point of the area [0,10000] |
| dwEndPointY | y-coordinate of the end point of the area [0,10000] |

**See also:**

[NETDEV\_PTZSelZoomIn](#_拉框放大.缩小_1)

### Structure of info about PTZ auto-guard

|  |
| --- |
| typedef struct tagNETDEVPTZGuardInfo  {  BOOL bEnabled;  UINT32 udwMode;  UINT32 udwParam;  UINT32 udwTime；  BYTE byRes[32];  }NETDEV\_PTZ\_GUARD\_INFO\_S, \*LPNETDEV\_PTZ\_GUARD\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Whether auto guard is enabled |
| udwMode | PTZ auto guard modes See [NETDEV\_PTZ\_GUARD\_MODE\_E](#_云台守望模式). |
| udwParam | Parameter. When Mode is 0, Param means a preset ID (starts from 1). When Mode is 1, Param means a patrol route index (starts from 0). When Mode is 2, Param means a recorded patrol route index (starts from 0). |
| udwTime | Auto guard time (unit: second). The pan/tilt unit starts auto-guard after being idle for the set period. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_PTZGetGuard\_Other](#_获取、设置云台守望信息)

### Absolute coordinates movement

|  |
| --- |
| typedef struct tagNETDEVPTZAbsoluteMove  {  FLOAT fPanTiltX;  FLOAT fPanTiltY;  FLOAT fZoomX;  }NETDEV\_PTZ\_ABSOLUTE\_MOVE\_S, \*LPNETDEV\_PTZ\_ABSOLUTE\_MOVE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| fPanTiltX | Absolute horizontal coordinate |
| fPanTiltY | Absolute vertical coordinate |
| fZoomX | Absolute zoom ratio |

**See also:**

[NETDEV\_PTZAbsoluteMove](#_云台绝对坐标转动)

### PTZ status

|  |
| --- |
| typedef struct tagNETDEVPTZStatus  {  FLOAT fPanTiltX;  FLOAT fPanTiltY;  FLOAT fZoomX;  [NETDEV\_PTZ\_MOVE\_STATUS\_E](#_聚焦状态) enPanTiltStatus;  [NETDEV\_PTZ\_MOVE\_STATUS\_E](#_聚焦状态) enZoomStatus;  }NETDEV\_PTZ\_STATUS\_S, \*LPNETDEV\_PTZ\_STATUS\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| fPanTiltX | Absolute horizontal coordinate |
| fPanTiltY | Absolute vertical coordinate |
| fZoomX | Absolute zoom ratio |
| enPanTiltStatus | PTZ status |
| enZoomStatus | Focus status |

**See also:**

[NETDEV\_PTZGetStatus](#_获取云台状态)

### Structure of PTZ recorded patrol route

|  |
| --- |
| typedef struct tagNETDEVPtzTrackinfo  {  INT32 dwTrackNum;  CHAR aszTrackName[NETDEV\_TRACK\_CRUISE\_MAXNUM][NETDEV\_LEN\_64];  }NETDEV\_PTZ\_TRACK\_INFO\_S, \*LPNETDEV\_PTZ\_TRACK\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwTrackNum | Number of exiting patrol routes |
| aszTrackName | Patrol route name |

**See also:**

[NETDEV\_PTZGetTrackCruise](#_获取云台轨迹巡航路径_1)

### PTZ recorded patrol route list

|  |
| --- |
| typedef struct tagNETDEVPtzTrackListV30  {  INT32 dwTrackNum;  [NETDEV\_PTZ\_TRACK\_INFO\_V30\_S](#_云台轨迹巡航路径结构体_1) astTrackInfo[NETDEV\_TRACK\_CRUISE\_MAXNUM];  }NETDEV\_PTZ\_TRACK\_LIST\_V30\_S, \*LPNETDEV\_PTZ\_TRACK\_LIST\_V30\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwTrackNum | Number of exiting patrol routes |
| astTrackInfo | Patrol route name |

**See also:**

[NETDEV\_PTZGetTrackCruise\_V30](#_获取云台轨迹巡航路径)

### Structure of PTZ recorded patrol route

|  |
| --- |
| typedef struct tagNETDEVPtzTrackInfoV30  {  UINT32 udwTrackID;  CHAR szTrackName[NETDEV\_LEN\_64];  BYTE byRes[64];  }NETDEV\_PTZ\_TRACK\_INFO\_V30\_S, \*LPNETDEV\_PTZ\_TRACK\_INFO\_V30\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTrackID | Patrol route ID |
| szTrackName | Patrol route name |
| byRes | Reserved field. |

**See also:**

[NETDEV\_PTZTrackCruise\_V30](#_云台轨迹巡航操作)

### Structure of info about status of PTZ recorded patrol

|  |
| --- |
| typedef struct tagNETDEVPTZTrackStatusInfo  {  INT32 dwStatus;  INT32 dwPatrolID;  }NETDEV\_PTZ\_TRACK\_STATUS\_INFO\_S, \*LPNETDEV\_PTZ\_TRACK\_STATUS\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwStatus | Channel’s patrol status See [NETDEV\_PTZ\_TRACK\_STATUS\_E](#_巡航状态枚举). |
| dwPatrolID | ID of recorded or preset patrol route |

**See also:**

[NETDEV\_PTZGetTrackStatus](#_获取指定通道的巡航状态)

### Structure of all PTZ presets

|  |
| --- |
| typedef struct tagstNETDEVPtzAllPresets  {  INT32 dwSize;  [NETDEV\_PTZ\_PRESET\_S](#_云台预置位信息结构体) astPreset[NETDEV\_MAX\_PRESET\_NUM];  }NETDEV\_PTZ\_ALLPRESETS\_S, \*LPNETDEV\_PTZ\_ALLPRESETS\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Total number of presets |
| astPreset | Structure of preset info |

**See also:**

[NETDEV\_GetPTZPresetList](#_获取云台预置位列表)

### Structure of PTZ preset info

|  |
| --- |
| typedef struct tagstNETDEVPtzPreset  {  INT32 dwPresetID;  CHAR szPresetName[NETDEV\_LEN\_32];  }NETDEV\_PTZ\_PRESET\_S, \*LPNETDEV\_PTZ\_PRESET\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwPresetID | Preset ID |
| szPresetName | Preset name |

**See also:**

[NETDEV\_PTZ\_ALLPRESETS\_S](#_获取云台预置位列表)

### Structure of detailed info about PTZ preset patrol routes

|  |
| --- |
| typedef struct tagNETDEVCruiseInfo  {  INT32 dwCuriseID;  CHAR szCuriseName[NETDEV\_LEN\_32];  INT32 dwSize;  [NETDEV\_CRUISE\_POINT\_S](#_云台预置位巡航轨迹点结构体) astCruisePoint[NETDEV\_MAX\_CRUISEPOINT\_NUM];  }NETDEV\_CRUISE\_INFO\_S, \*LPNETDEV\_CRUISE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwCuriseID | Route ID |
| szCuriseName | Patrol route name |
| dwSize | Number of presets in the patrol route |
| astCruisePoint | Info about points in the patrol route |

**See also:**

[NETDEV\_PTZCruise](#_云台预置位巡航操作_1)

### Structure of PTZ presets in preset patrol route

|  |
| --- |
| typedef struct tagNETDEVCruisePoint  {  INT32 dwPresetID;  INT32 dwStayTime;  INT32 dwSpeed;  INT32 dwID;  }NETDEV\_CRUISE\_POINT\_S, \*LPNETDEV\_CRUISE\_POINT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwPresetID | Preset ID |
| dwStayTime | Stay time |
| dwSpeed | Rotation speed. Range: [1-10] |
| dwID | Patrol action ID |

**See also:**

[NETDEV\_CRUISE\_INFO\_S](#_云台预置位巡航路径详细信息结构体)

### Structure of PTZ preset patrol route list

|  |
| --- |
| typedef struct tagNETDEVCruiseList  {  INT32 dwSize;  [NETDEV\_CRUISE\_INFO\_S](#_云台预置位巡航路径详细信息结构体) astCruiseInfo[NETDEV\_MAX\_CRUISEROUTE\_NUM];  }NETDEV\_CRUISE\_LIST\_S, \*LPNETDEV\_CRUISE\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Number of patrol routes |
| astCruiseInfo | Patrol route info |

**See also:**

[NETDEV\_PTZGetCruise](#_获取云台预置位巡航路径)

### Structure of PTZ direction info

|  |
| --- |
| typedef struct tagNETDEVPTZOrientationInfo  {  INT32 dwDirection;  BYTE byRes[252];  }NETDEV\_PTZ\_ORIENTATION\_INFO\_S, \*LPNETDEV\_PTZ\_ORIENTATION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwDirection | PTZ direction info See [NETDEV\_PTZ\_DIRECTION\_E](#_云台方位). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_PTZCalibrate](#_云台标定)

### Structure of PTZ lens angle

|  |
| --- |
| typedef struct tagNETDEVPTZLensAngle  {  FLOAT fHorizontalAngle;  FLOAT fVerticalAngle;  BYTE byRes[256];  }NETDEV\_PTZ\_LENSANGLE\_S, \*LPNETDEV\_PTZ\_LENSANGLE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| fHorizontalAngle | Horizontal field of view: [0.180] |
| fVerticalAngle | Vertical field of view: [0.180] |
| byRes | Reserved field. |

**See also:**

[NETDEV\_PTZGetLensAngle](#_获取指定视频输入通道的云台镜头当前视角值)

### PTZ latitude and longitude info unit

|  |
| --- |
| typedef struct tagNETDEVPTZPtPositionInfo  {  FLOAT fLongitude;  FLOAT fLatitude;  BYTE byRes[128];  }NETDEV\_PTZ\_PT\_POSITION\_INFO\_S, \*LPNETDEV\_PTZ\_PT\_POSITION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| fLongitude | Longitude (pan movement). Range: [0.00, 360.00]. Unit: degree. Precision: 2 decimal places |
| fLatitude | Latitude (tilt movement) |
| byRes | Reserved field. |

**See also:**

[NETDEV\_GetPTZAbsolutePTInfo](#_获取云台的经纬度信息)

### Structure of log search criteria

|  |
| --- |
| typedef struct tagNETDEVFindLogCond  {  INT32 dwMainType;  INT32 dwSubType;  INT64 tBeginTime;  INT64 tEndTime;  INT32 dwLimitNum;  INT32 dwOffset;  }NETDEV\_FIND\_LOG\_COND\_S, \*LPNETDEV\_FIND\_LOG\_COND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwMainType | Main log type. See [NETDEV\_LOG\_MAIN\_TYPE\_E](#_日志主类型枚举). |
| dwSubType | Sub log type. See [NETDEV\_LOG\_SUB\_TYPE\_E](#_日志子类型). |
| tBeginTime | Start time |
| tEndTime | End time |
| dwLimitNum | Number of results returned each time. |
| dwOffset | Sequence number from which the search begins. Starts from 0. |

**See also:**

[NETDEV\_FindLogInfoList](#_查找日志列表)

### Structure of log info

|  |
| --- |
| typedef struct tagNETDEVOptLogInfo  {  INT64 tTime;  INT32 dwMainType;  INT32 dwSubType;  INT32 dwChannelID;  CHAR szUserName[NETDEV\_NAME\_MAX\_LEN];  CHAR szUserAddr[NETDEV\_IPADDR\_STR\_MAX\_LEN];  CHAR szDetailInfo[NETDEV\_LEN\_256];  CHAR szOperObject[NETDEV\_LEN\_256];  UINT32 udwOperResult;  CHAR szDevName[NETDEV\_LEN\_256];  CHAR szOrgName[NETDEV\_LEN\_256];  BYTE bRes[256];  }NETDEV\_LOG\_INFO\_S,\*LPNETDEV\_LOG\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| tTime | Log time |
| dwMainType | Main log type. See [NETDEV\_LOG\_MAIN\_TYPE\_E](#_日志主类型枚举). |
| dwSubType | Sub log type. See [NETDEV\_LOG\_SUB\_TYPE\_E](#_日志子类型). |
| dwChannelID | Log source (channel ID) |
| szUserName | Username |
| szUserAddr | User’s IP address |
| szDetailInfo | Detailed info. Applicable to NVR only. |
| szOperObject | Operation object. Applicable to VMS only. |
| udwOperResult | Operation result. Applicable to VMS only. |
| szDevName | Device name |
| szOrgName | Organization name |
| bRes | Reserved field. |

**See also:**

[NETDEV\_FindNextLogInfo](#_逐个查找日志信息)

### Structure of alarm info search criteria

|  |
| --- |
| typedef struct tagNETDEVAlarmFindConds  {  INT32 dwChannelID;  INT64 tBeginTime;  INT64 tEndTime;  BYTE byRes[128];  }NETDEV\_FIND\_ALARM\_COND\_S, \*LPNETDEV\_FIND\_ALARM\_COND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| tBeginTime | Start time |
| tEndTime | End time |
| bRes | Reserved field. |

**See also:**

[NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表)

### Structure of alarm info

|  |
| --- |
| typedef struct tagNETDEVFindAlarmInfo  {  INT64 tAlarmTime;  INT32 dwAlarmType;  BYTE byRes[256];  }NETDEV\_FIND\_ALARM\_INFO\_S, \*LPNETDEV\_FIND\_ALARM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| tAlarmTime | Alarm time |
| dwAlarmType | Alarm type. See [NETDEV\_FIND\_ALARM\_TYPE\_E](#_告警类型枚举). |
| bRes | Reserved field. |

**See also:**

[NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息)

### Alarm log search criteria list

|  |
| --- |
| typedef struct tagstNETDEVAlarmLogCondList  {  INT32 dwPageRow;  INT32 dwFirstRow;  INT32 dwCondSize;  [NETDEV\_QUERY\_INFO\_S](#_告警日志查询条件结构体) astCondition[NETDEV\_LOG\_QUERY\_COND\_NUM];  }NETDEV\_ALARM\_LOG\_COND\_LIST\_S, \*LPNETDEV\_ALARM\_LOG\_COND\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwPageRow | Maximum number of rows per page. |
| dwFirstRow | Sequence number of the first row in paginated search |
| dwCondSize | Number of search criteria |
| astCondition | Rvalue of search criteria |

**See also:**

[NETDEV\_FindAlarmLogList](#_获取告警日志列表), [NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录)

### Structure of alarm log search criteria

|  |
| --- |
| typedef struct tagstNETDEVQueryInfo  {  INT32 dwQueryType;  INT32 dwLogicFlag;  CHAR szConditionData[NETDEV\_CODE\_STR\_MAX\_LEN];  }NETDEV\_QUERY\_INFO\_S, \*LPNETDEV\_QUERY\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwQueryType | Search criteria type. See [NETDEV\_QUERYCOND\_TYPE\_E](#_告警日志查询条件枚举). |
| dwLogicFlag | Logic of search criteria. See [NETDEV\_QUERYCOND\_LOGICTYPE\_E](#_查询条件逻辑类型枚举). |
| szConditionData | Rvalue of search criteria |

**See also:**

[NETDEV\_ALARM\_LOG\_COND\_LIST\_S](#_告警日志查询信息结构体)

### Structure of alarm log info

|  |
| --- |
| typedef struct tagstNETDEVAlarmLogInfo  {  INT32 dwAlarmID;  INT32 dwAlarmType;  INT32 dwAlarmSubType;  INT32 dwAlarmLevel;  INT32 dwServerID;  INT32 dwDevID;  INT32 dwChannelID;  CHAR szAlarmSrc[NETDEV\_NAME\_MAX\_LEN];  INT64 tAlarmTime;  BOOL bAlarmChecked;  CHAR szAlarmCheckUser[NETDEV\_USERNAME\_LEN];  INT64 tAlarmCheckTime;  CHAR szAlarmCheckDesc[NETDEV\_DESCRIBE\_MAX\_LEN];  INT32 dwAlarmLinkType;  BOOL IsAlarmSnapExisted;  INT32 dwAlarmSrcBelong;  CHAR szAlarmDetail[NETDEV\_DESCRIBE\_MAX\_LEN];  INT32 dwHasRelatedData;  BYTE byRes[244];  }NETDEV\_ALARM\_LOG\_INFO\_S, \*LPNETDEV\_ALARM\_LOG\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwAlarmID | Alarm ID |
| dwAlarmType | Alarm type. See [NETDEV\_ALARM\_TYPE\_E](#_告警类型枚举_1). |
| dwAlarmSubType | Alarm sub type. See [NETDEV\_ALARM\_SUBTYPE\_E](#_告警子类型枚举). |
| dwAlarmLevel | Alarm level. 0: Critical 1: Major 2: Minor 3: Warning 4: Alert |
| dwServerID | Server |
| dwDevID | Device IP |
| dwChannelID | Channel ID |
| szAlarmSrc | Alarm source info |
| tAlarmTime | Alarm occurrence time (unit: second, UTC format) |
| bAlarmChecked | Whether alarm is acknowledged. 0: No 1: Yes |
| szAlarmCheckUser | User who acknowledged the alarm |
| tAlarmCheckTime | Alarm acknowledgement time (unit: second, UTC format) |
| szAlarmCheckDesc | Remarks given when acknowledging an alarm |
| dwAlarmLinkType | Alarm linkage type |
| IsAlarmSnapExisted | Whether an alarm snapshot exists. 0: No 1: Yes |
| dwAlarmSrcBelong | Alarm type. See [NETDEV\_ALARM\_SRC\_BELONG\_TYPE\_E](#_告警所属类型枚举) |
| szAlarmDetail | Alarm details. |
| dwHasRelatedData | Whether alarm has related data. 0: No 1: Yes |
| byRes | Reserved field. |

**See also:**

[NETDEV\_FindNextAlarmLog](#_逐个获取告警日志信息)

### Structure of person list

|  |
| --- |
| typedef struct tagNETDEVPersonInfoList  {  UINT32 udwNum;  [LPNETDEV\_PERSON\_INFO\_S](#_人员信息结构体) pstPersonInfo;  BYTE byRes[128];  }NETDEV\_PERSON\_INFO\_LIST\_S,\*LPNETDEV\_PERSON\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of members in the person library |
| pstPersonInfo | Person list. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_AddPersonInfo](#_新增指定的人员信息)

### Structure of person info

|  |
| --- |
| typedef struct tagNETDEVPersonInfo  {  UINT32 udwPersonID;  UINT32 udwLastChange;  CHAR szPersonName[NETDEV\_LEN\_256];  UINT32 udwGender;  CHAR szBirthday[NETDEV\_LEN\_64];  [NETDEV\_REGION\_INFO\_S](#_成员地区信息结构体) stRegionInfo;  UINT32 udwTimeTemplateNum;  [LPNETDEV\_PERSON\_TIME\_TEMPLATE\_INFO\_S](#_时间模板相关信息结构体) pstTimeTemplateList;  UINT32 udwIdentificationNum;  [NETDEV\_IDENTIFICATION\_INFO\_S](#_成员证件信息结构体) stIdentificationInfo[NETDEV\_LEN\_8];  UINT32 udwImageNum;  [NETDEV\_IMAGE\_INFO\_S](#_人脸图片信息列表结构体) stImageInfo[NETDEV\_LEN\_8];  UINT32 udwReqSeq;  BOOL bIsMonitored;  UINT32 udwBelongLibNum;  UINT32 \*pudwBelongLibList;  UINT32 udwCustomNum;  [LPNETDEV\_CUSTOM\_VALUE\_S](#_自定义属性信息结构体) pstCustomValueList;  CHAR szTelephone[NETDEV\_LEN\_64];  CHAR szAddress[NETDEV\_LEN\_256];  UINT32 udwCardNum;  UINT32 udwFingerprintNum;  UINT32 udwType;  [NETDEV\_STAFF\_INFO\_S](#_员工信息结构体) stStaff;  [NETDEV\_VISITOR\_INFO\_S](#_访客信息结构体) stVisitor;  CHAR szDesc[NETDEV\_LEN\_480];  CHAR szPersonCode[NETDEV\_LEN\_16];  CHAR szRemarks[NETDEV\_LEN\_64];  BYTE byRes[176];  }NETDEV\_PERSON\_INFO\_S,\*LPNETDEV\_PERSON\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPersonID | Person ID |
| udwLastChange | Time when person info was last modified |
| szPersonName | Person name. Range: [1, 63] |
| udwGender | Gender. See [NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举). |
| szBirthday | Birth date (YYYYMMDD). Range: [1,31] |
| stRegionInfo | Member region info |
| udwTimeTemplateNum | Number of time templates |
| pstTimeTemplateList | Time template info. Memory needs to be allocated dynamically. |
| udwIdentificationNum | Number of IDs. Range: [0, 6] |
| stIdentificationInfo | Member ID info |
| udwImageNum | Number of face images. Range: [0, 6] |
| stImageInfo | Face image list |
| udwReqSeq | Sequence number of request data. This field is carried in the returned result. Required only in batch adding. |
| bIsMonitored | Whether monitoring is configured. Required when using the GET method. Applicable to VMS only. |
| udwBelongLibNum | Number of libraries containing the member. Applicable to VMS only. |
| pudwBelongLibList | ID of libraries containing the member. Memory needs to be allocated dynamically. Applicable to VMS only. |
| udwCustomNum | Number of custom attributes. Max: 5. Applicable to VMS only. |
| pstCustomValueList | Custom attribute value list. Optional when Num is 0. Memory needs to be allocated dynamically. |
| szTelephone | Telephone number. String length: [1, 64] |
| szAddress | Address. String length: [1, 64] |
| udwCardNum | Number of access control cards. Range: [0, 6]. Required when using the GET method. |
| udwFingerprintNum | Number of fingerprints. Range: [0, 10] |
| udwType | Person type. 0: Staff 1: Visitor 2: Stranger |
| stStaff | Staff info |
| stVisitor | Visitor info |
| szDesc | Remarks. String length range: [1, 128]. |
| szPersonCode | Person ID. Fill in with student ID or employee ID. Range: [1, 15]. Applicable to PTS. |
| szRemarks | Remarks. Range: [1-63]. Applicable to PTS. |
| byRes | Reserved field. |

### Structure of member’s region info

|  |
| --- |
| typedef struct tagNETDEVRegionInfo  {  CHAR szNation[NETDEV\_LEN\_128];  CHAR szProvince[NETDEV\_LEN\_128];  CHAR szCity[NETDEV\_LEN\_128];  BYTE byRes[256];  }NETDEV\_REGION\_INFO\_S, \*LPNETDEV\_REGION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szNation | Nationality. Range: [1-63] |
| szProvince | Province. Range: [1-63] |
| szCity | City. Range: [1-63] |
| byRes | Reserved field. |

### Structure of time template info

|  |
| --- |
| typedef struct tagNETDEVPersonTimeTemplateInfo  {  UINT32 udwBeginTime;  UINT32 udwEndTime;  UINT32 udwIndex;  BYTE byRes[128];  }NETDEV\_PERSON\_TIME\_TEMPLATE\_INFO\_S,\*LPNETDEV\_PERSON\_TIME\_TEMPLATE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwBeginTime | Start time of validity period of time template. Fill in 0 if not configured. |
| udwEndTime | End time of validity period of time template. Fill in 4294967295(0xFFFFFFFF) if not configured. |
| udwIndex | Time template index |
| byRes | Reserved field. |

### Structure of member’s ID info

|  |
| --- |
| typedef struct tagNETDEVIdentificationInfo  {  UINT32 udwType;  CHAR szNumber[NETDEV\_LEN\_128];  BYTE byRes[32];  }NETDEV\_IDENTIFICATION\_INFO\_S,\*LPNETDEV\_IDENTIFICATION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | ID type. See [NETDEV\_ID\_TYPE\_E](#_证件类型枚举). |
| szNumber | ID number. Range: [1, 127] |
| byRes | Reserved field. |

### Structure of face image list

|  |
| --- |
| typedef struct tagNETDEVImageInfo  {  UINT32 udwFaceID;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stFileInfo;  UINT32 udwModelStatus;  BYTE byRes[124];  }NETDEV\_IMAGE\_INFO\_S,\*LPNETDEV\_IMAGE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwFaceID | Face image ID |
| stFileInfo | File info |
| udwModelStatus | Modeling status. See [NETDEV\_MODEL\_STATUS\_E](#_建模状态枚举). |
| byRes | Reserved field. |

### Structure of file info

|  |
| --- |
| typedef struct tagNETDEVFileInfo  {  CHAR szName[NETDEV\_LEN\_64];  UINT32 udwSize;  UINT32 dwFileType;  UINT32 udwLastChange;  CHAR \*pcData;  CHAR szUrl[NETDEV\_LEN\_512];  BYTE byRes[128];  }NETDEV\_FILE\_INFO\_S,\*LPNETDEV\_FILE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szName | Filename. Range: [1, 16] |
| udwSize | File size. [data or image size obtained through szurl (Base64-encoded)] |
| dwFileType | File type. See [NETDEV\_FILE\_TYPE\_E](#_文件信息枚举). |
| udwLastChange | Last modified time (unit: second, UTC format) |
| pcData | File data. Base64-encoded. Memory needs to be allocated dynamically according to udwSize. |
| szUrl | Image URL. Length range: [0,256] |
| byRes | Reserved field. |

### Structure of custom attribute info

|  |
| --- |
| typedef struct tagNETDEVCustomValue  {  UINT32 udwID;  CHAR szValue[NETDEV\_FACE\_MEMBER\_CUSTOM\_LEN];  UINT32 udwModelStatus;  BYTE byRes[124];  }NETDEV\_CUSTOM\_VALUE\_S,\*LPNETDEV\_CUSTOM\_VALUE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Sequence number of customer attribute name. Starts from 0. |
| szValue | Value of custom attributes. Range: [1,63] |
| udwModelStatus | Modeling status. See [NETDEV\_MODEL\_STATUS\_E](#_建模状态枚举). |
| byRes | Reserved field. |

### Structure of staff info

|  |
| --- |
| typedef struct tagNETDEVStaffInfo  {  CHAR szNumber[NETDEV\_LEN\_32];  CHAR szBirthday[NETDEV\_LEN\_32];  UINT32 udwDeptID;  CHAR szDeptName[NETDEV\_LEN\_256];  BYTE byRes[256];  }NETDEV\_STAFF\_INFO\_S,\*LPNETDEV\_STAFF\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szNumber | Member ID, string length: [1, 16] |
| szBirthday | Birth date, string length: [1,31] |
| udwDeptID | Department ID |
| szDeptName | Department name, optional when adding a department. String length range: [1, 64] |
| byRes | Reserved field. |

### Structure of visitor info

|  |
| --- |
| typedef struct tagNETDEVVisitorInfo  {  UINT32 udwVisitorCount;  CHAR szCompany[NETDEV\_LEN\_256];  UINT32 udwIntervieweeID;  BYTE byRes[256];  }NETDEV\_VISITOR\_INFO\_S,\*LPNETDEV\_VISITOR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwVisitorCount | Number of visitors |
| szCompany | Company of the visitor. String length range: [1, 64] |
| udwIntervieweeID | ID of the visited |
| byRes | Reserved field. |

### Structure of person info result list

|  |
| --- |
| typedef struct tagNETDEVPersonResultList  {  UINT32 udwNum;  [LPNETDEV\_PERSON\_LIST\_S](#_人员信息执行结果结构体) pstPersonList;  BYTE byRes[128];  }NETDEV\_PERSON\_RESULT\_LIST\_S,\*LPNETDEV\_PERSON\_RESULT\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of people. |
| pstPersonList | Person info result list. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_AddPersonInfo](#_新增指定的人员信息)

### Structure of person info processing result

|  |
| --- |
| typedef struct tagNETDEVPersonList  {  UINT32 udwPersonID;  UINT32 udwFaceNum;  [NETDEV\_FACE\_INFO\_S](#_人脸信息结果结构体) stFaceInfo[NETDEV\_LEN\_8];  UINT32 udwReqseq;  BYTE byRes[128];  }NETDEV\_PERSON\_LIST\_S,\*LPNETDEV\_PERSON\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPersonID | Person ID |
| udwFaceNum | Number of faces. Max 6 each batch. |
| stFaceInfo | Face info result list |
| udwReqseq | Sequence number of request data. Applicable to VMS only. |
| byRes | Reserved field. |

### Structure of face info processing result

|  |
| --- |
| typedef struct tagNETDEVFaceInfo  {  UINT32 udwFaceID;  UINT32 udwResultCode;  BYTE byRes[128];  }NETDEV\_FACE\_INFO\_S,\*LPNETDEV\_FACE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwFaceID | Person ID |
| udwResultCode | Status code for processing result. See [NETDEV\_PERSON\_RESULT\_CODE\_E](#_人脸处理结果状态码枚举). |
| byRes | Reserved field. |

### Structure of monitoring task info

|  |
| --- |
| typedef struct tagNETDEVMonitorInfo  {  UINT32 udwID;  [NETDEV\_MONITION\_RULE\_INFO\_S](#_布控任务配置信息结构体) stMonitorRuleInfo;  UINT32 udwLinkStrategyNum;  [LPNETDEV\_LINKAGE\_STRATEGY\_S](#_告警联动配置信息结构体) pstLinkStrategyList;  [NETDEV\_VIDEO\_WEEK\_PLAN\_S](#_计划（周）配置结构体) stWeekPlan;  [NETDEV\_MONITOR\_DEFENCE\_INFO\_S](#_布防信息结构体) stMonitorDefenceInfo;  BYTE byRes[250];  }NETDEV\_MONITION\_INFO\_S,\*LPNETDEV\_MONITION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Sequence number of face monitoring task. Not returned when adding a monitoring task. |
| stMonitorRuleInfo | Configuration information of face monitoring task. |
| udwLinkStrategyNum | Number of alarm linkage strategies |
| pstLinkStrategyList | Configuration of alarm linkage strategy. Memory needs to be allocated dynamically. |
| stWeekPlan | Arming schedule of face monitoring task. Applicable to NVR and IPC only. |
| stMonitorDefenceInfo | Arming information. Applicable to PTS, VMS only. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_AddPersonMonitorInfo](#_新增单个人脸布控任务)

### Structure of info about monitoring task configuration

|  |
| --- |
| typedef struct tagNETDEVMonitorRuleInfo  {  BOOL bEnabled;  CHAR szName[NETDEV\_FACE\_MONITOR\_RULE\_NAME\_LEN];  CHAR szReason[NETDEV\_FACE\_MONITOR\_RULE\_REASON\_LEN];  UINT32 udwLibNum;  UINT32 audwLibList[NETDEV\_LEN\_32];  UINT32 udwMonitorType;  UINT32 udwMultipleValue;  UINT32 udwMonitorReason;  CHAR szMatchSucceedMsg[NETDEV\_LEN\_512];  CHAR szMatchFailedMsg[NETDEV\_LEN\_512];  UINT32 udwMemberNum;  [NETDEV\_MEMBER\_INFO\_S](#_人脸/车辆成员信息列表结构体) stMemberInfo[NETDEV\_LEN\_64];  UINT32 udwChannelNum;  UINT32 \*pudwMonitorChlIDList;  UINT32 udwDevNum;  UINT32 audwMonitorDevIDList[NETDEV\_LEN\_16];  UINT32 udwMonitorRuleType;  BYTE byRes[240];  }NETDEV\_MONITION\_RULE\_INFO\_S,\*LPNETDEV\_MONITION\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Whether monitoring task is enabled. |
| szName | Name of monitoring task. |
| szReason | Cause of monitoring |
| udwLibNum | Number of libraries corresponding to the task. Max: 16. |
| audwLibList | Library ID list |
| udwMonitorType | Monitoring alarm type 0: Match alarm 1: Not match alarm |
| udwMultipleValue | Confidence threshold for 1:N face comparison |
| udwMonitorReason | Causes of vehicle monitoring: 0: robbed vehicle, 1: stolen vehicle, 2: suspect vehicle, 3: traffic violation, 4: emergency control |
| szMatchSucceedMsg | Message for a match |
| szMatchFailedMsg | Message for a mismatch |
| udwMemberNum | Number of members (0-32). Applicable to NVR only. |
| stMemberInfo | Member list. |
| udwChannelNum | Number of monitoring channels used by a monitoring task. Required when getting detailed info about a monitoring task. |
| pudwMonitorChlIDList | List of monitoring channels used by a monitoring task. Memory needs to be dynamically according to udwChannelNum. |
| udwDevNum | Number of monitoring devices used by a monitoring task. Max: 4. Applicable to VMS only. |
| audwMonitorDevIDList | List of monitoring devices used by a monitoring task. Memory needs to be allocated dynamically according to DevNum. Applicable to VMS only. |
| udwMonitorRuleType | Analysis type of monitoring task, 0: Local analysis, 1: Frontend analysis |
| byRes | Reserved field. |

### Structure of face/vehicle member list

|  |
| --- |
| typedef struct tagNETDEVMemberInfo  {  UINT32 udwMemberID;  CHAR szMemberName[NETDEV\_LEN\_260];  UINT32 udwStatus;  BYTE byRes[124];  }NETDEV\_MEMBER\_INFO\_S,\*LPNETDEV\_MEMBER\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwMemberID | Face/vehicle member ID |
| szMemberName | Face/vehicle member name. Range: [1,63] |
| udwStatus | Member synchronization status. See [NETDEV\_PERSON\_RESULT\_CODE\_E](#_人脸处理结果状态码枚举). |
| byRes | Reserved field. |

### Structure of alarm linkage configuration info

|  |
| --- |
| typedef struct tagLinkageStrategy  {  UINT32 udwType;  [NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表结构体) stLintageActions;  BYTE byRes[512];  }NETDEV\_LINKAGE\_STRATEGY\_S, \*LPNETDEV\_LINKAGE\_STRATEGY\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | Alarm type. See [NETDEV\_PERSON\_COMPARE\_RESULT\_TYPE\_E](#_告警联动配置信息枚举). |
| stLintageActions | Linkage actions of face monitoring task |
| byRes | Reserved field. |

### Structure of linkage actions list of monitoring task

|  |
| --- |
| typedef struct tagNETDEVLinkageActionList  {  UINT32 udwNum;  [NETDEV\_LINKAGE\_ACTION\_INFO\_S](#_联动动作列表信息结构体) stActionInfo[NETDEV\_MAX\_LINK\_ACTION\_NUM];  BYTE byRes[256];  }NETDEV\_LINKAGE\_ACTION\_LIST\_S,\*LPNETDEV\_LINKAGE\_ACTION\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of linkage action list |
| stActionInfo | Info about linkage action list |
| byRes | Reserved field. |

### Structure of info about linkage action list

|  |
| --- |
| typedef struct tagNETDEVLinkageActionInfo  {  UINT32 udwActID;  [NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S](#_联动使能参数结构体) stEnabledInfo;  [NETDEV\_OUTPUT\_SWITCH\_ACT\_PARAM\_INFO\_S](#_联动开关量输出结构体) stOutputSwitchActParamInfo;  [NETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S](#_通道联动信息结构体) stChannelActParamInfo;  [NETDEV\_PRESET\_ACT\_PARAM\_INFO\_S](#_联动云台预置位列表结构体) stPresetActParamInfo;  BYTE byRes[512];  }NETDEV\_LINKAGE\_ACTION\_INFO\_S,\*LPNETDEV\_LINKAGE\_ACTION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwActID | Action ID. See [NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举). |
| stEnabledInfo | Enable or disable linkage action parameters. Applicable to alarm-triggered buzzer, email and pop-up window. |
| stOutputSwitchActParamInfo | Link alarm output |
| stChannelActParamInfo | Trigger NVR preview, recording, and snapshot |
| stPresetActParamInfo | Trigger PTZ preset |
| byRes | Reserved field. |

### Structure of channel linkage info

|  |
| --- |
| typedef struct tagNETDEVChannelActParamInfo  {  UINT32 udwNum;  INT32 adwChannelID[NETDEV\_CHANNEL\_MAX];  }NETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S,\*LPNETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of channels |
| adwChannelID | Channel ID list |

### Structure of linkage enablement

|  |
| --- |
| typedef struct tagNETDEVEnabledActParamInfo  {  BOOL bEnabled;  BYTE byRes[64];  }NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S, \*LPNETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Enablement flag. |
| byRes | Reserved field. |

### Structure of linked PTZ preset list

|  |
| --- |
| typedef struct tagNETDEVPresetActParamInfo  {  UINT32 udwNum;  [NETDEV\_CHANNEL\_PRESET\_S](#_联动云台预置位信息结构体) stChannelPreset[NETDEV\_CHANNEL\_MAX];  }NETDEV\_PRESET\_ACT\_PARAM\_INFO\_S,\*LPNETDEV\_PRESET\_ACT\_PARAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of linkage actions |
| stChannelPreset | List of linked PTZ presets |

### Structure of info about linked PTZ presets

|  |
| --- |
| typedef struct tagNETDEVChannelPreset  {  INT32 dwChannelID;  INT32 dwPresetID;  BYTE byRes[128];  }NETDEV\_CHANNEL\_PRESET\_S,\*LPNETDEV\_CHANNEL\_PRESET\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| dwPresetID | Preset ID |
| byRes | Reserved field. |

### Structure of linked alarm output

|  |
| --- |
| typedef struct tagNETDEVOutputSwitchActParamInfo  {  UINT32 udwNum;  [NETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S](#_输出开关量的逻辑报警状态(手动告警)结构体) astOutputAlarmStatusInfo[NETDEV\_MAX\_ALARM\_OUT\_NUM];  } NETDEV\_OUTPUT\_SWITCH\_ACT\_PARAM\_INFO\_S,\* LPNETDEV\_OUTPUT\_SWITCH\_ACT\_PARAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of linked alarm outputs |
| astOutputAlarmStatusInfo | List of linked alarm outputs |

### Structure of logical alarm status of alarm output (manual alarm)

|  |
| --- |
| typedef struct tagNETDEVOutputSwitchesAlarmStatus  {  INT32 dwBooleanId;  INT32 dwChannelId;  INT32 enAlarmStatus;  } NETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S, \*LPNETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwBooleanId | Boolean ID |
| dwChannelId | Channel ID. For the device it is 0. |
| enAlarmStatus | Alarm output status. See [NETDEV\_RELAYOUTPUT\_STATE\_E](#_开关量状态枚举). |

### Structure of schedule configuration (weekly)

|  |
| --- |
| typedef struct tagNETDEVVideoPlanWeek  {  BOOL bEnabled;  UINT32 udwDayNum;  [NETDEV\_VIDEO\_DAY\_PLAN\_S](#_计划（天）配置结构体) astDayPlan[NETDEV\_MAX\_DAY\_NUM];  BYTE byRes[32];  }NETDEV\_VIDEO\_WEEK\_PLAN\_S, \*LPNETDEV\_VIDEO\_WEEK\_PLAN\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Whether arming schedule is enabled. Only IPC supports the enable/disable switch; for VMS and NVR arming schedule is enabled by default. See [NETDEV\_DEFENCE\_PLAN\_ENABLE\_E](#_布防计划使能枚举). |
| udwDayNum | Number of scheduled days.  NVR: Max 8 (7 days of a week plus holiday). IPC: Max 7 (7 days of a week). |
| astDayPlan | Arming schedule list of every day in a week. |
| byRes | Reserved field. |

### Schedule of schedule configuration (daily)

|  |
| --- |
| typedef struct tagNETDEVVideoDayPlan  {  UINT32 udwIndex;  UINT32 udwSectionNum;  [NETDEV\_VIDEO\_TIME\_SECTION\_S](#_时间段配置结构体) astTimeSection[NETDEV\_MAX\_TIME\_SECTION\_NUM];  }NETDEV\_VIDEO\_DAY\_PLAN\_S, \*LPNETDEV\_VIDEO\_DAY\_PLAN\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwIndex | Index of days in a week. See [NETDEV\_LAPI\_WEEK\_INFO\_E](#_星期枚举). |
| udwSectionNum | Number of time periods per day. NVR: Max 8; IPC: Max 4. |
| astTimeSection | Arming period configuration |

### Structure of time period configuration

|  |
| --- |
| typedef struct tagNETDEVVideoTimeSection  {  CHAR szBeginTime[NETDEV\_LEN\_64];  CHAR szEndTime[NETDEV\_LEN\_64];  UINT32 udArmingType;  }NETDEV\_VIDEO\_TIME\_SECTION\_S, \*LPNETDEV\_VIDEO\_TIME\_SECTION\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szBeginTime | Start time |
| szEndTime | End time |
| udArmingType | Arming type. See [NETDEV\_ARMING\_TYPE\_E](#_告警布防类型枚举). |

### Structure of arming info

|  |
| --- |
| typedef struct tagNETDEVMonitorDefenceInfo  {  INT64 tBegin;  INT64 tEnd;  UINT32 udwTimeTemplateID;  BYTE byRes[128];  }NETDEV\_MONITOR\_DEFENCE\_INFO\_S, \*LPNETDEV\_MONITOR\_DEFENCE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| tBegin | Validity period start time of time template (Unix timestamp) |
| tEnd | Validity period end time of time template (Unix timestamp) |
| udwTimeTemplateID | Time template index. Fill in 0 if not configured. |
| byRes | Reserved field. |

### Structure of monitoring list returned after adding monitoring

|  |
| --- |
| typedef struct tagNETDEVMonitorResultInfo  {  UINT32 udwChannelNum;  [LPNETDEV\_MONITION\_CHL\_INFO\_S](#_添加布控返回的布控信息结构体) pstMonitorChlInfos;  BYTE byRes[250]; /  }NETDEV\_MONITOR\_RESULT\_INFO\_S,\*LPNETDEV\_MONITOR\_RESULT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwChannelNum | Actual number of channels added successfully for monitoring. Value assignment is needed to indicate the requested memory size. |
| pstMonitorChlInfos | Returned monitoring channel list. Memory needs to be requested by the upper layer and should not be less than the number of channels. |
| byRes | Reserved field. |

### Structure of monitoring info returned after adding monitoring

|  |
| --- |
| typedef struct tagNETDEVMonitorChlInfo  {  UINT32 udwChannelID;  UINT32 udwResultCode;  UINT32 udwMonitorID;  }NETDEV\_MONITION\_CHL\_INFO\_S,\*LPNETDEV\_MONITION\_CHL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwChannelID | Corresponding channel ID of monitoring task. For IPC and VMS this field is not returned. |
| udwResultCode | Result code for face processing. See [NETDEV\_PERSON\_RESULT\_CODE\_E](#_人脸处理结果状态码枚举). |
| udwMonitorID | Corresponding monitoring task ID of the channel |

### Structure of person library info

|  |
| --- |
| typedef struct tagNETDEVLibInfo  {  UINT32 udwID;  CHAR szName[NETDEV\_LEN\_256];  UINT32 udwType;  UINT32 udwPersonNum;  UINT32 udwFaceNum;  UINT32 udwMemberNum;  UINT32 udwLastChange;  CHAR szBelongIndex[NETDEV\_LEN\_256];  BOOL bIsMonitored;  BYTE byRes[128];  }NETDEV\_LIB\_INFO\_S,\*LPNETDEV\_LIB\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Library ID |
| szName | Library name. Range: [1,63] |
| udwType | Person library type. See [NETDEV\_PEOPLE\_LIB\_TYPE\_E](#_人员库类型枚举). |
| udwPersonNum | Total number of persons in the library |
| udwFaceNum | Total number of face images in the library |
| udwMemberNum | Total number of members in the library |
| udwLastChange | Last time library information is modified. |
| szBelongIndex | Index that uniquely identifies the library |
| bIsMonitored | Whether monitoring is configured. Required when getting library info. |
| byRes | Reserved field. |

### Structure of person library list

|  |
| --- |
| typedef struct tagNETDEVPersonLibList  {  UINT32 udwNum;  [LPNETDEV\_LIB\_INFO\_S](#_人员库信息结构体) pstLibInfo;  BYTE byRes[128];  }NETDEV\_PERSON\_LIB\_LIST\_S,\*LPNETDEV\_PERSON\_LIB\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of libraries in the device |
| pstLibInfo | Library info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Structure of flag for deleting a library

|  |
| --- |
| typedef struct tagNETDEVDeleteDBFlagInfo  {  BOOL bIsDeleteMember;  UINT32 udwDevID;  BYTE byRes[124];  }NETDEV\_DELETE\_DB\_FLAG\_INFO\_S,\*LPNETDEV\_DELETE\_DB\_FLAG\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bIsDeleteMember | Whether to delete members in library. 0: No 1: Yes |
| udwDevID | Device ID (Only VMS supports deleting face libraries). |
| byRes | Reserved field. |

### Structure of info about capacity of all person libraries

|  |
| --- |
| typedef struct tagNETDEVPersonLibCapList  {  UINT32 udwMaxPerpleMun;  UINT32 udwFreePerpleNum;  UINT32 udwMaxLibNum;  UINT32 udwFreeLibNum;  UINT32 udwNum;  [NETDEV\_PERSON\_LIB\_CAP\_INFO\_S](#_人脸库容量信息结构体) stLibCapInfoList[NETDEV\_LEN\_16];  BYTE byRes[256];  }NETDEV\_PERSON\_LIB\_CAP\_LIST\_S, \*LPNETDEV\_PERSON\_LIB\_CAP\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwMaxPerpleMun | Total library capacity (unit: 1000 persons) |
| udwFreePerpleNum | Remaining capacity (unit: person) |
| udwMaxLibNum | Maximum number of libraries allowed |
| udwFreeLibNum | Number of remaining libraries that can be created |
| udwNum | Number of libraries created. Range: [0,16] |
| stLibCapInfoList | Capacity information of individual libraries |
| byRes | Reserved field. |

### Structure of info about capacity of a face library

|  |
| --- |
| typedef struct tagNETDEVPersonLibCapInfo  {  UINT32 udwLibID;  UINT32 udwCapacity;  BYTE byRes[128];  }NETDEV\_PERSON\_LIB\_CAP\_INFO\_S, \*LPNETDEV\_PERSON\_LIB\_CAP\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwLibID | Library ID |
| udwCapacity | Library capacity info (unit: person) |
| byRes | Reserved field. |

### Structure of person info search criteria

|  |
| --- |
| typedef struct tagNETDEVPersonQueryInfo  {  UINT32 udwNum;  [LPNETDEV\_QUERY\_INFO\_S](#_告警日志查询条件结构体) pstQueryInfos;  UINT32 udwLimit;  UINT32 udwOffset;  BYTE byRes[256];  }NETDEV\_PERSON\_QUERY\_INFO\_S, \*LPNETDEV\_PERSON\_QUERY\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of search conditions |
| pstQueryInfos | Search condition list. Optional when Num is 0. |
| udwLimit | Search limit. Max: 20. |
| udwOffset | Sequence number from where the search starts. Starts from 0. |
| byRes | Reserved field. |

### Structure of basic info returned for a batch search

|  |
| --- |
| typedef struct tagNETDEVBatchOperateBasicInfo  {  UINT32 udwTotal;  UINT32 udwOffset;  UINT32 udwNum;  BYTE byRes[128];  }NETDEV\_BATCH\_OPERATE\_BASIC\_S,\*LPNETDEV\_BATCH\_OPERATE\_BASIC\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTotal | Quantity |
| udwOffset | Sequence number from where the search starts |
| udwNum | Total number of search results |
| byRes | Reserved field. |

### Structure of list returned for batch deleting face monitoring

|  |
| --- |
| typedef struct tagNETDEVBatchOperateList  {  UINT32 udwNum;  UINT32 udwStatus;  [LPNETDEV\_BATCH\_OPERATOR\_INFO\_S](#_批量操作信息结构体) pstBatchList;  BYTE byRes[128];  }NETDEV\_BATCH\_OPERATOR\_LIST\_S,\*LPNETDEV\_BATCH\_OPERATOR\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Batch processed number |
| udwStatus | Result status |
| pstBatchList | Batch operation info. Max 2000. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Structure of batch operation info

|  |
| --- |
| typedef struct tagNETDEVBatchOperatorInfo  {  UINT32 udwReqSeq;  UINT32 udwResultCode;  UINT32 udwID;  CHAR szName[NETDEV\_LEN\_260];  BYTE byRes[128];  }NETDEV\_BATCH\_OPERATOR\_INFO\_S,\*LPNETDEV\_BATCH\_OPERATOR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwReqSeq | Sequence number of request data |
| udwResultCode | Returned error code. See [NETDEV\_PERSON\_MONITOR\_OPT\_RES\_CODE\_E](#_人脸布控操作结果错误码枚举). |
| udwID | ID |
| szName | Member name. Length range: [1,63] |
| byRes | Reserved field. |

### Criteria for searching alarm snapshot URL

|  |
| --- |
| typedef struct tagNETDEVAlarmSnapShotCond  {  INT32 dwChannelID;  INT64 tAlarmTime;  [NETDEV\_ALARM\_SNAPSHOT\_TYPE\_E](#_告警抓图类型枚举) enAlarmType;  BYTE byRes[244];  }NETDEV\_ALARM\_SNAPSHOT\_COND\_S,\*LPNETDEV\_ALARM\_SNAPSHOT\_COND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID or alarm output ID. The parameter is channel ID\*100 + alarm output ID. For a device the channel ID is 0. |
| tAlarmTime | Alarm time |
| enAlarmType | Alarm type |
| byRes | Reserved field. |

**See also:**

[NETDEV\_FindAlarmSnapShotURL](#_获取告警联动抓图URL列表)

### Alarm snapshot image info

|  |
| --- |
| typedef struct tagNETDEVAlarmSnapShotPicInfo  {  CHAR szURL[NETDEV\_MAX\_URL\_LEN];  CHAR szName[NETDEV\_LEN\_64];  INT32 dwSize;  BYTE byRes[256];  }NETDEV\_ALARM\_SNAPSHOT\_PIC\_S,\*LPNETDEV\_ALARM\_SNAPSHOT\_PIC\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szURL | Image URL |
| szName | Image name |
| dwSize | Image size |
| byRes | Reserved field. |

**See also:**

[NETDEV\_FindNextAlarmSnapShotURL](#_逐个获取查找到的人员信息)

### Structure of snapshot image info

|  |
| --- |
| typedef struct tagNETDEVSnapShotFileInfo  {  CHAR szURL[NETDEV\_MAX\_URL\_LEN];  BOOL bSaveLocal;  CHAR szFileName[NETDEV\_LEN\_260];  CHAR\* pcBuffer;  INT32 dwSize;  INT32 dwCaptureMode;  BYTE byRes[252];  }NETDEV\_PIC\_FILE\_INFO\_S, \*LPNETDEV\_PIC\_FILE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szURL | Image URL |
| bSaveLocal | Whether image is saved to local. True: Yes False: Write to buffer |
| szFileName | Local path and name of the saved image |
| pcBuffer | Buffer for saving images. Memory needs to be requested by user according to dwSize. |
| dwSize | Buffer size |
| dwCaptureMode | Format of saved image. See [NETDEV\_PICTURE\_FORMAT\_E](#_抓图图片格式枚举). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SaveSnapShotFile](#_保存告警抓图)

### Structure of point coordinates

|  |
| --- |
| typedef struct tagNETDEVPointS  {  INT32 dwPointX;  INT32 dwPointY;  }NETDEV\_POINT\_S, \*LPNETDEV\_POINT\_S; |

|  |  |
| --- | --- |
| Parameter | Description |
| dwPointX | x-coordinate (‱). Range: [0-10000] |
| dwPointY | y-coordinate (‱). Range: [0-10000] |

**See also:**

[NETDEV\_SetMouseMoveMode](#_鼠标操作模式), [NETDEV\_GetFishEyeParam](#_获取鱼眼矫正参数)

### Structure of manual recording

|  |
| --- |
| typedef struct tagNETDEVManualRecordcfg  {  INT32 dwChannelID;  [NETDEV\_RECORD\_TYPE\_E](#_录像类型枚举) enRecordType;  BYTE byRes[64];  }NETDEV\_MANUAL\_RECORD\_CFG\_S, \*LPNETDEV\_MANUAL\_RECORD\_CFG\_S; |

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| enRecordType | Recording type. See [NETDEV\_RECORD\_TYPE\_E](#_录像类型枚举). |
| byRes | Reserved field |

**See also:**

[NETDEV\_GetManualRecordStatus](#_获取手动录像状态), [NETDEV\_StartManualRecord](#_开启手动录像), [NETDEV\_StopManualRecord](#_停止手动录像)

### Basic device info

|  |
| --- |
| typedef struct tagNETDEVDeviceBasicInfo  {  CHAR szDevModel[NETDEV\_LEN\_64];  CHAR szSerialNum[NETDEV\_LEN\_64];  CHAR szFirmwareVersion[NETDEV\_LEN\_64];  CHAR szMacAddress[NETDEV\_LEN\_64];  CHAR szDeviceName[NETDEV\_LEN\_64];  CHAR szManufacturer[NETDEV\_LEN\_64];  BYTE byRes[384];  }NETDEV\_DEVICE\_BASICINFO\_S, \*LPNETDEV\_DEVICE\_BASICINFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szDevModel | Device model |
| szSerialNum | Hardware serial number |
| szFirmwareVersion | Firmware version |
| szMacAddress | MAC address (IPv4) |
| szDeviceName | Device name |
| szManufacturer | Manufacturer info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### NTP parameters

|  |
| --- |
| typedef struct tagNETDEVSystemNTPInfo  {  BOOL bSupportDHCP;  [NETDEV\_SYSTEM\_IPADDR\_S](#_地址) stAddr;  }NETDEV\_SYSTEM\_NTP\_INFO\_S, \*LPNETDEV\_SYSTEM\_NTP\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bSupportDHCP | Whether DHCP is supported |
| stAddr | NTP information |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Struct of address

|  |
| --- |
| typedef struct tagSysemIPAddr  {  INT32 eIPType;  CHAR szIPAddr[NETDEV\_LEN\_132];  }NETDEV\_SYSTEM\_IPADDR\_S, \*LPNETDEV\_SYSTEM\_IPADDR\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| eIPType | Protocol type. See [NETDEV\_HOSTTYPE\_E](#_IP协议类型). |
| szIPAddr | IP address |

**See also:**

NETDEV\_SYSTEM\_NTP\_INFO\_S

### NTP list

|  |
| --- |
| typedef struct tagNETDEVSystemNTPInfoList  {  INT64 ulNum;  [NETDEV\_SYSTEM\_IPADDR\_INFO\_S](#_系统IP地址信息) astNTPServerInfoList[NETDEV\_NTP\_SERVER\_LIST\_NUM];  BYTE byRes[128];  }NETDEV\_SYSTEM\_NTP\_INFO\_LIST\_S, \*LPNETDEV\_SYSTEM\_NTP\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| ulNum | Number of NTP servers |
| astNTPServerInfoList | NTP server list. Currently only one NTP server is allowed. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Video stream information of a channel

|  |
| --- |
| typedef struct tagNETDEVVideoStreamInfo  {  [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举) enStreamType;  INT32 bEnableFlag;  INT32 dwHeight;  INT32 dwWidth;  INT32 dwFrameRate;  INT32 dwBitRate;  [NETDEV\_VIDEO\_CODE\_TYPE\_E](#_视频编码格式枚举) enCodeType;  [NETDEV\_VIDEO\_QUALITY\_E](#_视频图像质量) enQuality;  INT32 dwGop;  BOOL bConstantBitRate;  BYTE byRes[28];  }NETDEV\_VIDEO\_STREAM\_INFO\_S, \*LPNETDEV\_VIDEO\_STREAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| enStreamType | Stream index |
| bEnableFlag | Enable or disable |
| dwHeight | Video resolution - height |
| dwWidth | Video resolution - width |
| dwFrameRate | Video encoding frame rate |
| dwBitRate | Bit rate |
| enCodeType | Video encoding format |
| enQuality | Image quality |
| dwGop | I frame interval |
| bConstantBitRate | Whether is CBR. 0: VBR 1: CBR |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Video stream list

|  |
| --- |
| typedef struct tagNETDEVVideoStreamInfoList  {  UINT32 udwNum;  [NETDEV\_VIDEO\_STREAM\_INFO\_EX\_S](#_视频流信息) astVideoStreamInfoList[NETDEV\_LEN\_16];  }NETDEV\_VIDEO\_STREAM\_INFO\_LIST\_S,\*LPNETDEV\_VIDEO\_STREAM\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of video streams |
| astVideoStreamInfoList | Video stream list |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Video stream info

|  |
| --- |
| typedef struct tagNETDEVVideoStreamInfoLapi  {  BOOL bEnabled;  UINT32 udwStreamID;  UINT32 udwMainStreamType;  [NETDEV\_VIDEO\_ENCODE\_INFO\_S](#_视频编码参数信息) stVideoEncodeInfo;  }NETDEV\_VIDEO\_STREAM\_INFO\_EX\_S,\*LPNETDEV\_VIDEO\_STREAM\_INFO\_EX\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Enable or disable video stream encoding |
| udwStreamID | Stream index. See [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举). |
| udwMainStreamType | Main stream type. See [NETDEV\_MAIN\_STREAM\_TYPE\_E](#_主码流类型). |
| stVideoEncodeInfo | Video encoding parameter info |

**See also:**

[NETDEV\_VIDEO\_STREAM\_INFO\_LIST\_S](#_视频流信息列表)

### Video encoding parameter info

|  |
| --- |
| typedef struct tagNETDEVVideoEncodeInfo  {  BOOL bEnableSVCMode;  UINT32 udwEncodeFormat;  UINT32 udwWidth;  UINT32 udwHeight;  UINT32 udwBitrate;  UINT32 udwBitrateType;  UINT32 udwFrameRate;  UINT32 udwGopType;  UINT32 udwIFrameInterval;  UINT32 udwImageQuality;  UINT32 udwSmoothLevel;  UINT32 udwSmartEncodeMode;  }NETDEV\_VIDEO\_ENCODE\_INFO\_S,\*LPNETDEV\_VIDEO\_ENCODE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnableSVCMode | Enable or disable SVC. 0: Disable 1: Enable |
| udwEncodeFormat | Video encoding format info. See [NETDEV\_VIDEO\_CODE\_TYPE\_E](#_视频编码格式枚举). |
| udwWidth | Image width |
| udwHeight | Image height |
| udwBitrate | Bit rate |
| udwBitrateType | Bit rate type. See [NETDEV\_BIT\_RATE\_TYPE\_E](#_码率模式类型). |
| udwFrameRate | Frame rate |
| udwGopType | GOP mode. See [NETDEV\_GOP\_TYPE\_E](#_GOP类型). |
| udwIFrameInterval | I frame interval. The range depends on capability. |
| udwImageQuality | Image quality. Range: [0 9]. 9 is the highest quality. |
| udwSmoothLevel | Stream smooth level. Range: [1,9]. 1 is the lowest smooth level. |
| udwSmartEncodeMode | Smart encoding mode. See [NETDEV\_SMART\_ENCODE\_MODE\_E](#_图像扩展编码模式类型). |

**See also:**

[NETDEV\_VIDEO\_STREAM\_INFO\_EX\_S](#_视频编码参数信息)

### Video collection capability

|  |
| --- |
| typedef struct tagNETDEVVideoModeInfo  {  UINT32 udwWidth;  UINT32 udwHeight;  UINT32 udwFrameRate;  }NETDEV\_VIDEO\_MODE\_INFO\_S,\*LPNETDEV\_VIDEO\_MODE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwWidth | Image width |
| udwHeight | Image height |
| udwFrameRate | Image frame rate |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Basic attributes of OSD

|  |
| --- |
| typedef struct tagNETDEVOsdCfgInfo  {  [NETDEV\_OSD\_TIME\_S](#_OSD时间配置信息) stTimeOSD;  [NETDEV\_OSD\_TEXT\_OVERLAY\_S](#_OSD字符叠加信息_1) stNameOSD;  INT16 wTextNum;  [NETDEV\_OSD\_TEXT\_OVERLAY\_S](#_OSD字符叠加信息_1) astTextOverlay[NETDEV\_OSD\_TEXTOVERLAY\_NUM];  }NETDEV\_VIDEO\_OSD\_CFG\_S, \*LPNETDEV\_VIDEO\_OSD\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stTimeOSD | Time OSD info |
| stNameOSD | Name OSD |
| wTextNum | Number of text OSDs |
| astTextOverlay | OSD text |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### OSD time configuration info

|  |
| --- |
| typedef struct tagNETDEVOsdTime  {  BOOL bEnableFlag;  BOOL bWeekEnableFlag;  [NETDEV\_AREA\_SCOPE\_S](#_区域) stAreaScope;  UINT32 udwTimeFormat;  UINT32 udwDateFormat;  }NETDEV\_OSD\_TIME\_S, \*LPNETDEV\_OSD\_TIME\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnableFlag | Whether time OSD is enabled. TRUE: Yes FALSE: No |
| bWeekEnableFlag | Whether week is displayed (reserved) |
| stAreaScope | Coordinates of area |
| udwTimeFormat | Time format of time OSD. See [NETDEV\_OSD\_TIME\_FORMAT\_CAP\_E](#_OSD时间格式能力集枚举). |
| udwDateFormat | Time format of date OSD. See [NETDEV\_OSD\_DATE\_FORMAT\_CAP\_E](#_OSD日期格式能力集). |

**See also:**

[NETDEV\_VIDEO\_OSD\_CFG\_S](#_通道OSD的基本属性信息)

### OSD text

|  |
| --- |
| typedef struct tagNETDEVOsdTextOverlay  {  BOOL bEnableFlag;  [NETDEV\_AREA\_SCOPE\_S](#_区域) stAreaScope;  CHAR szOSDText[NETDEV\_OSD\_TEXT\_MAX\_LEN];  BYTE byRes[4];  }NETDEV\_OSD\_TEXT\_OVERLAY\_S, \*LPNETDEV\_OSD\_TEXT\_OVERLAY\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnableFlag | Enable or disable OSD text. TRUE: Yes FALSE: No |
| stAreaScope | Coordinates of OSD text area |
| szOSDText | OSD text name string |
| byRes | Reserved field. |

**See also:**

[NETDEV\_VIDEO\_OSD\_CFG\_S](#_通道OSD的基本属性信息)

### Area

|  |
| --- |
| typedef struct tagNETDEVAreaScope  {  INT32 dwLocateX;  INT32 dwLocateY;  }NETDEV\_AREA\_SCOPE\_S, \*LPNETDEV\_AREA\_SCOPE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwLocateX | Coordinates of vertex x [0,10000] |
| dwLocateY | Coordinates of vertex y [0,10000] |

**See also:**

[NETDEV\_OSD\_TIME\_S](#_OSD时间配置信息), [NETDEV\_OSD\_TEXT\_OVERLAY\_S](#_OSD字符叠加信息_1)

### OSD configuration list

|  |
| --- |
| typedef struct tagNETDEVChlOsdCfgList  {  INT32 dwSize;  [NETDEV\_CHL\_OSD\_CFG\_S](#_通道OSD配置) stChlOsdCfg[NETDEV\_OSD\_MAX\_NUM\_EX];  }NETDEV\_CHL\_OSD\_CFG\_LIST\_S, \*LPNETDEV\_CHL\_OSD\_CFG\_LIST\_S； |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Number of OSDs |
| stChlOsdCfg | OSD configuration |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### OSD configuration

|  |
| --- |
| typedef struct tagNETDEVChlOsdCfg  {  INT32 dwOsdID;  [NETDEV\_OSD\_TEXT\_TYPE\_E](#_OSD类型) enOsdTextType;  [NETDEV\_AREA\_SCOPE\_S](#_区域) stAreaScope;  CHAR szTextInfo[NETDEV\_OSD\_TEXT\_MAX\_LEN];  }NETDEV\_CHL\_OSD\_CFG\_S, \*LPNETDEV\_CHL\_OSD\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwOsdID | OSD ID |
| enOsdTextType | OSD type |
| stAreaScope | OSD coordinates |
| szTextInfo | OSD contents |

**See also:**

[NETDEV\_CHL\_OSD\_CFG\_LIST\_S](#_通道OSD配置列表)

### OSD content style

|  |
| --- |
| typedef struct tagNETDEVOsdContentStyle  {  UINT32 udwFontStyle;  UINT32 udwFontSize;  UINT32 udwColor;  UINT32 udwDateFormat;  UINT32 udwTimeFormat;  UINT32 audwFontAlignList[NETDEV\_LEN\_8];  UINT32 udwMargin;  }NETDEV\_OSD\_CONTENT\_STYLE\_S,\*LPNETDEV\_OSD\_CONTENT\_STYLE\_S； |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwFontStyle | Font style. See [NETDEV\_OSD\_FONT\_STYLE\_E](#_OSD字体形式枚举). |
| udwFontSize | Font size. See [NETDEV\_OSD\_FONT\_SIZE\_E](#_OSD字体大小枚举). |
| udwColor | Color |
| udwDateFormat | Date format. See [NETDEV\_OSD\_DATE\_FORMAT\_E](#_日期格式). |
| udwTimeFormat | Time format. See [NETDEV\_OSD\_TIME\_FORMAT\_E](#_OSD时间格式枚举). |
| audwFontAlignList | Text alignment in area. 8 areas. Applicable to IPC. See [NETDEV\_OSD\_ALIGN\_E](#_OSD区域内对齐枚举). |
| udwMargin | Number of characters with margin. Applicable to IPC. See [NETDEV\_OSD\_MIN\_MARGIN\_E](#_OSD字体大小枚举_1). |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### All alarm input info

|  |
| --- |
| typedef struct tagNETDEVAlarmOutputList  {  INT32 dwSize;  [NETDEV\_ALARM\_OUTPUT\_INFO\_S](#_告警开关量输出信息) astAlarmOutputInfo[NETDEV\_MAX\_ALARM\_OUT\_NUM];  } NETDEV\_ALARM\_OUTPUT\_LIST\_S, \*LPNETDEV\_ALARM\_OUTPUT\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Number of alarm inputs |
| astAlarmOutputInfo | Alarm input configuration info |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Alarm output info

|  |
| --- |
| typedef struct tagNETDEVAlarmOutputInfo  {  CHAR szName[NETDEV\_LEN\_64];  INT32 dwChancelId;  INT32 enDefaultStatus;  INT32 dwDurationSec;  INT32 dwOutputNum;  } NETDEV\_ALARM\_OUTPUT\_INFO\_S, \*LPNETDEV\_ALARM\_OUTPUT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szName | Alarm output name |
| dwChancelId | Channel ID |
| enDefaultStatus | Default alarm output status. See [NETDEV\_BOOLEAN\_MODE\_E](#_输入开关量运行模式). |
| dwDurationSec | Alarm duration (unit: second) |
| dwOutputNum | Alarm output sequence number |

**See also:**[NETDEV\_ALARM\_OUTPUT\_LIST\_S](#_所有告警开关量输入信息)

### Link alarm input

|  |
| --- |
| typedef struct tagNETDEVTriggerAlarmOutput  {  CHAR szName[NETDEV\_LEN\_64];  [NETDEV\_RELAYOUTPUT\_STATE\_E](#_开关量状态枚举) enOutputState;  } NETDEV\_TRIGGER\_ALARM\_OUTPUT\_S, \*LPNETDEV\_TRIGGER\_ALARM\_OUTPUT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szName | Alarm input name |
| enOutputState | Trigger status |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### All alarm input info

|  |
| --- |
| typedef struct tagNETDEVAlarmInputInfoList  {  INT32 dwSize;  [NETDEV\_ALARM\_INPUT\_INFO\_S](#_告警开关量输入信息) astAlarmInputInfo[NETDEV\_MAX\_ALARM\_IN\_NUM];  }NETDEV\_ALARM\_INPUT\_LIST\_S, \*LPNETDEV\_ALARM\_INPUT\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Number of alarm inputs |
| astAlarmInputInfo | Alarm input configuration info |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Alarm input info

|  |
| --- |
| typedef struct tagNETDEVAlarmInputInfo  {  CHAR szName[NETDEV\_LEN\_64];  }NETDEV\_ALARM\_INPUT\_INFO\_S, \*LPNETDEV\_ALARM\_INPUT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szName | Alarm input name |

**See also:**

[NETDEV\_ALARM\_INPUT\_LIST\_S](#_所有告警开关量输入信息_1)

### Logical alarm status list of alarm output (manual alarm)

|  |
| --- |
| typedef struct tagNETDEVOutputSwitchesAlarmStatusList  {  INT32 dwSize;  [NETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S](#_输出开关量的逻辑报警状态(手动告警))  astOutputAlarmStatusInfo[NETDEV\_MAX\_ALARM\_OUT\_NUM];  BYTE bRes[128];  }NETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_LIST\_S,  \*LPNETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_LIST\_S |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Number of alarm output channels |
| astOutputAlarmStatusInfo | Alarm output status list |
| bRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Logical alarm status of alarm output (manual alarm)

|  |
| --- |
| typedef struct tagNETDEVOutputSwitchesAlarmStatus  {  INT32 dwBooleanId;  INT32 dwChannelId;  INT32 enAlarmStatus;  }NETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S, \*LPNETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwBooleanId | Alarm output ID |
| dwChannelId | Channel ID. For the device it is 0. |
| enAlarmStatus | Alarm output status. See [NETDEV\_RELAYOUTPUT\_STATE\_E](#_开关量状态枚举). |

**See also:**

[NETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_LIST\_S](#_输出开关量的逻辑报警状态列表(手动告警))

### Structure of logical alarm status of triggering or clearing switch output (manual alarm)

|  |
| --- |
| typedef struct tagNETDEVOutputSwitchesManualAlarmInfo  {  INT32 dwSize;  INT32 dwIDList[NETDEV\_MAX\_ALARM\_OUT\_NUM];  [NETDEV\_MANUAL\_ALARM\_CMD\_E](#_控制手动告警开关量状态命令) dwAlarmAction;  BYTE bRes[128];  }NETDEV\_OUTPUT\_SWITCH\_MANUAL\_ALARM\_INFO\_S, \*LPNETDEV\_OUTPUT\_SWITCH\_MANUAL\_ALARM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Number of alarm outputs to set |
| dwIDList | List of alarm output IDs to set |
| dwAlarmAction | Alarm action |
| bRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Alarm input configuration list

|  |
| --- |
| typedef struct tagNETDEVInputSwitchesInfoList  {  INT32 udwNum;  [NETDEV\_INPUT\_SWITCH\_INFO\_S](#_输入开关量告警信息) astInputSwitchesInfo[NETDEV\_LEN\_1024];  }NETDEV\_INPUT\_SWITCH\_INFO\_LIST\_S, \*LPNETDEV\_INPUT\_SWITCH\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of alarm inputs |
| astInputSwitchesInfo | Alarm input configuration info |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Alarm input info

|  |
| --- |
| typedef struct tagNETDEVInputSwitchesInfo  {  INT32 dwChancelId;  INT32 dwInputSwitchIndex;  CHAR szName[NETDEV\_LEN\_64];  CHAR szGBID[NETDEV\_LEN\_32];  INT32 dwRunMode;  INT32 dwEnabled;  INT32 dwVideoResID;  BYTE bRes[124];  }NETDEV\_INPUT\_SWITCH\_INFO\_S, \*LPNETDEV\_INPUT\_SWITCH\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChancelId | Channel ID |
| dwInputSwitchIndex | Alarm output index |
| szName | Alarm input name |
| szGBID | GB resource ID. Applicable to IPC only. |
| dwRunMode | Operation mode. 1: Normally open 2: Normally closed |
| dwEnabled | Enable or disable alarm. 0: Disable 1: Enable |
| dwVideoResID | Corresponding video channel ID. Applicable to VMS only. |
| bRes | Reserved field. |

**See also:**

[NETDEV\_INPUT\_SWITCH\_INFO\_LIST\_S](#_输入开关量告警配置信息列表)

### Image settings

|  |
| --- |
| typedef struct tagNETDEVImagingSetting  {  INT32 dwContrast; /\* contrast \*/  INT32 dwBrightness; /\* brightness \*/  INT32 dwSaturation; /\* saturation \*/  INT32 dwSharpness; /\* sharpness \*/  BYTE byRes[252]; /\* Reserved field \*/  }NETDEV\_IMAGE\_SETTING\_S, \*LPNETDEV\_IMAGE\_SETTING\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwContrast | Contrast |
| dwBrightness | Brightness |
| dwSaturation | Saturation |
| dwSharpness | Brightness |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Image exposure parameters

|  |
| --- |
| typedef struct tagNETDEVImagingExposure  {  UINT32 udwMode;  INT32 dwCompensationLevel;  UINT32 udwHLCSensitivity;  [NETDEV\_IRIS\_INFO\_S](#_光圈信息) stIrisInfo;  [NETDEV\_SHUTTER\_INFO\_S](#_快门信息) stShutterInfo;  [NETDEV\_GAIN\_INFO\_S](#_增益信息_1) stGainInfo;  [NETDEV\_WIDE\_DYNAMIC\_INFO\_S](#_宽动态信息_1) stWideDynamicInfo;  [NETDEV\_METERING\_INFO\_S](#_测光信息_1) stMeteringInfo;  [NETDEV\_DAY\_NIGHT\_INFO\_S](#_昼夜模式信息_1) stDayNightInfo;  BYTE byRes[256];  }NETDEV\_IMAGE\_EXPOSURE\_S, \*LPNETDEV\_IMAGE\_EXPOSURE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwMode | Exposure mode. See [NETDEV\_EXPOSURE\_MODE\_E](#_曝光模式). |
| dwCompensationLevel | Exposure compensation level, valid when exposure mode is not "manual". Range: [-100,100]. Required when supported by image capability. |
| udwHLCSensitivity | HLC sensitivity, valid when the scene is Road HLC or Park HLC, range: [1,9]. Required if supported by image capability. |
| stIrisInfo | Iris info. Required if supported by image capability. |
| stShutterInfo | Shutter info. Required if supported by image capability. |
| stGainInfo | Gain info |
| stWideDynamicInfo | WDR info Required if supported by image capability. |
| stMeteringInfo | Metering info Valid when the scene is not Road HLC and Park HLC. Required if supported by image capability. |
| stDayNightInfo | Day/night mode info Required if supported by image capability. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Iris info

|  |
| --- |
| typedef struct tagNETDEVIrisInfo  {  UINT32 udwIris;  UINT32 udwMinIris;  UINT32 udwMaxIris;  BYTE byRes[128];  }NETDEV\_IRIS\_INFO\_S, \*LPNETDEV\_IRIS\_INFO\_S |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwIris | Iris. Optional in iris priority and manual exposure modes. Supported values: 160, 200, 240, 280, 340, 400, 480, 560, 680, 800, 960, 1100, 1600, 2200 |
| udwMinIris | Minimum iris. Valid in custom exposure mode. Must not be greater than the maximum iris. See iris capability. Required if supported by image capability. |
| udwMaxIris | Maximum iris. Valid in custom exposure mode. Must not be less than the minimum iris. See iris capability. Required if supported by image capability. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)

### Shutter info

|  |
| --- |
| typedef struct tagNETDEVShutterInfo  {  UINT32 udwShutterTime;  UINT32 udwMinShutterTime;  UINT32 udwMaxShutterTime;  UINT32 udwIsEnableSlowShutter;  UINT32 udwSlowestShutter;  BYTE byRes[128];  }NETDEV\_SHUTTER\_INFO\_S, \*LPNETDEV\_SHUTTER\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwShutterTime | Shutter time. See [NETDEV\_SHUTTER\_TIME\_RANGE\_E](#_快门时间支持的取值). Time unit: 0: ms 1: s |
| udwMinShutterTime | Minimum shutter time. See [NETDEV\_SHUTTER\_TIME\_RANGE\_E](#_快门时间支持的取值). |
| udwMaxShutterTime | Maximum shutter time. See [NETDEV\_SHUTTER\_TIME\_RANGE\_E](#_快门时间支持的取值). |
| udwIsEnableSlowShutter | Whether slow shutter is enabled Valid when not in iris priority mode. 0: No 1: Yes |
| udwSlowestShutter | Slowest shutter. Valid when slow shutter is enabled. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)

### Gain info

|  |
| --- |
| typedef struct tagNETDEVGainInfo  {  UINT32 udwGain;  UINT32 udwMinGain;  UINT32 udwMaxGain;  BYTE byRes[128];  }NETDEV\_GAIN\_INFO\_S, \*LPNETDEV\_GAIN\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwGain | Gain (unit: db). Valid in manual exposure mode. Range: [1,100] |
| udwMinGain | Minimum gain. Valid in custom exposure mode. Must not be greater than the maximum gain. Min: 1 |
| udwMaxGain | Maximum gain. Valid in custom exposure mode. Must not be LEDs than the minimum gain. Max: 100 |
| byRes | Reserved field. |

**See also:**

[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)

### WDR info

|  |
| --- |
| typedef struct tagNETDEVWideDynamicInfo  {  UINT32 udwWideDynamicMode;  UINT32 udwWideDynamicLevel;  UINT32 udwOpenSensitivity;  UINT32 udwCloseSensitivity;  UINT32 udwAntiFlicker;  BYTE byRes[128];  }NETDEV\_WIDE\_DYNAMIC\_INFO\_S, \*LPNETDEV\_WIDE\_DYNAMIC\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwWideDynamicMode | WDR mode. See [NETDEV\_WIDE\_DYNAMIC\_MODE\_E](#_测光控制模式). |
| udwWideDynamicLevel | WDR level. Valid when WDR is enabled and in one of the following exposure modes: Auto, Custom, Shutter Priority, Indoor 50Hz, Indoor 60Hz, or Low Motion Blur. Range: [1, 9]. |
| udwOpenSensitivity | Sensitivity of turning on WDR Valid when WDR is Auto. Range: [1, 9]. |
| udwCloseSensitivity | Sensitivity of turning off WDR Valid when WDR mode is Auto. Range: [1, 9]. |
| udwAntiFlicker | Suppress WDR stripes, which can eliminate stripes in images when enabled. 0: Disable 1: Enable |
| byRes | Reserved field. |

**See also:**

[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)

### Metering info

|  |
| --- |
| typedef struct tagNETDEVMeteringInfo  {  UINT32 udwMeteringMode;  UINT32 udwRefBrightness;  UINT32 udwHoldTime;  [NETDEV\_METERING\_AREA\_S](#_测光区域) stMeteringArea;  BYTE byRes[128];  }NETDEV\_METERING\_INFO\_S, \*LPNETDEV\_METERING\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwMeteringMode | Metering control mode. Valid when not in manual exposure mode. See [NETDEV\_DAY\_NIGHT\_MODE\_E](#_昼夜模式类型). |
| udwRefBrightness | Brightness of face. Valid in face metering mode. Range: [0, 100]. |
| udwHoldTime | Minimum duration. Valid in face metering mode. Unit: minute. Range: [0, 60]. |
| stMeteringArea | Metering area. Valid in evaluative metering or spot metering mode. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)

### Metering area

|  |
| --- |
| typedef struct tagNETDEVMeteringArea  {  [NETDEV\_AREA\_TOP\_LEFT\_S](#_左上角区域) stAreaTopLeft;  [NETDEV\_AREA\_BOT\_RIGHT\_S](#_右下角区域) stAreaBotRight;  BYTE byRes[128];  }NETDEV\_METERING\_AREA\_S, \*LPNETDEV\_METERING\_AREA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stAreaTopLeft | Upper left area |
| stAreaBotRight | Lower right area |
| byRes | Reserved field. |

**See also:**

[NETDEV\_METERING\_INFO\_S](#_测光信息_1)

### Upper left area

|  |
| --- |
| typedef struct tagNETDEVAreaTopLeft  {  UINT32 dwTopLeftX;  UINT32 dwTopLeftY;  BYTE byRes[128];  }NETDEV\_AREA\_TOP\_LEFT\_S, \*LPNETDEV\_AREA\_TOP\_LEFT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwTopLeftX | x-coordinate of the top left (scale). Evaluative metering range: [0, 100] |
| dwTopLeftY | y-coordinate of the top left (scale). Evaluative metering range: [0, 100] |
| byRes | Reserved field. |

**See also:**

[NETDEV\_METERING\_AREA\_S](#_测光区域)

### Lower right area

|  |
| --- |
| typedef struct tagNETDEVAreaBotRight  {  UINT32 dwBottomRightX;  UINT32 dwBottomRightY;  BYTE byRes[128];  }NETDEV\_AREA\_BOT\_RIGHT\_S, \*LPNETDEV\_AREA\_BOT\_RIGHT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwBottomRightX | x-coordinate of the top left (scale). Evaluative metering range: [0, 100] |
| dwBottomRightY | y-coordinate of the top left (scale). Evaluative metering range: [0, 100] |
| byRes | Reserved field. |

**See also:**

[NETDEV\_METERING\_AREA\_S](#_测光区域)

### Day/night mode info

|  |
| --- |
| typedef struct tagNETDEVDayNightInfo  {  UINT32 udwDayNightMode;  UINT32 udwDayNightSensitivity;  UINT32 udwDayNightTime;  BYTE byRes[128];  }NETDEV\_DAY\_NIGHT\_INFO\_S, \*LPNETDEV\_DAY\_NIGHT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwDayNightMode | Day/night mode. See [NETDEV\_DAY\_NIGHT\_MODE\_E](#_昼夜模式类型). |
| udwDayNightSensitivity | Day/night mode sensitivity. Valid when day/night mode is Auto. Range: [0,9]. Required if supported by image capability. |
| udwDayNightTime | Day/night mode switching time. Valid when day/night mode is Auto. Range: [3, 120]. Unit: second. Required if supported by image capability. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)

### Illuminator info

|  |
| --- |
| typedef struct tagNETDEVLampCtrlInfo  {  UINT32 udwEnabled;  UINT32 udwType;  UINT32 udwMode;  UINT32 udwNearLevel;  UINT32 udwMiddleLevel;  UINT32 udwFarLevel;  UINT32 udwSuperFarLevel;  UINT32 udwLaserAngle;  BYTE byRes[256];  }NETDEV\_LAMP\_CTRL\_INFO\_S, \*LPNETDEV\_LAMP\_CTRL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwEnabled | Turn on or off illuminator. 0: Off 1: On |
| udwType | Illuminator type. See [NETDEV\_LAMP\_TYPE\_E](#_补光灯类型). |
| udwMode | Illuminator control mode. See [NETDEV\_LAMP\_CTRL\_MODE\_E](#_补光灯支持的控制模式). |
| udwNearLevel | Near-illumination level. Valid when IR control mode is Manual or Manual-Always On. Required if supported by illuminator capability. Range: [0-1000]. A higher level means higher illumination intensity. |
| udwMiddleLevel | Mid-illumination level. Valid when IR control mode is Manual or Manual-Always On. Required if supported by illuminator capability. Range: [0-1000]. A higher level means higher illumination intensity. |
| udwFarLevel | Far-illumination level. Valid when IR control mode is Manual or Manual-Always On. Required if supported by illuminator capability. Range: [0-1000]. A higher level means higher illumination intensity. |
| udwSuperFarLevel | Super-far-illumination level. Valid when IR control mode is Manual or Manual-Always On. Required if supported by illuminator capability. Range: [0-1000]. A higher level means higher illumination intensity. |
| udwLaserAngle | Laser angle. Valid when Lighting Type is Laser and IR control mode is Manual or Manual-Always On. Required if supported by illuminator capability. See illuminator capability for the range. A smaller laser angle means intenser energy. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### White balance info

|  |
| --- |
| typedef struct tagNETDEVWhiteBalanceInfo  {  UINT32 udwMode;  INT32 dwRedOffset;  INT32 dwBlueOffset;  BYTE byRes[256];  }NETDEV\_WHITE\_BALANCE\_INFO\_S, \*LPNETDEV\_WHITE\_BALANCE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwMode | White balance mode. See [NETDEV\_WHITE\_BALANCE\_MODE\_E](#_白平衡模式). |
| dwRedOffset | Red offset. Configurable only in fine-tune mode (Mode is 1, 5, 8). |
| dwBlueOffset | Blue offset. Configurable only in fine-tune mode (Mode is 1, 5, 8). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Network configuration info

|  |
| --- |
| typedef struct tagNETDEVNetworkInterfaces  {  INT32 dwMTU;  BOOL bIPv4DHCP;  CHAR szIpv4Address[NETDEV\_LEN\_32];  CHAR szIPv4GateWay[NETDEV\_LEN\_32];  CHAR szIPv4SubnetMask[NETDEV\_LEN\_32];  BYTE byRes[480];  }NETDEV\_NETWORKCFG\_S, \*LPNETDEV\_NETWORKCFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwMTU | MTU value |
| bIPv4DHCP | DHCP (IPv4) |
| szIpv4Address | IP address (IPv4) |
| szIPv4GateWay | Gateway address (IPv4) |
| szIPv4SubnetMask | Subnet mask (IPv4) |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### VPN client info

|  |
| --- |
| typedef struct tagNETDEVVPNClientInfo  {  UINT32 udwEnabled;  UINT32 udwCompatibilityMode;  UINT32 udwAuthEnabled;  UINT32 udwEncryptEnabled;  UINT32 udwInterface;  CHAR szLoginName[NETDEV\_DOMAIN\_LEN];  CHAR szPIN[NETDEV\_LEN\_256];  UINT32 udwAddressType;  CHAR szAddress[NETDEV\_IPADDR\_STR\_MAX\_LEN];  UINT32 udwPort;  UINT32 udwUNPSuccessd;  CHAR szUNPShareIPAddr[NETDEV\_IPADDR\_STR\_MAX\_LEN];  CHAR szUNPShareMask[NETDEV\_IPADDR\_STR\_MAX\_LEN];  CHAR szUNPShareWayAddr[NETDEV\_IPADDR\_STR\_MAX\_LEN];  BYTE byRes[256];  }NETDEV\_VPN\_CLIENT\_INFO\_S, \*LPNETDEV\_VPN\_CLIENT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwEnabled | Enable or disable compatibility mode. 0: Disable 1: Enable |
| udwCompatibilityMode | Compatibility mode. 0: UNP1.0（unp）1: UNP2.0 (softvpn) |
| udwAuthEnabled | Enable or disable authentication. For UNP2.0 authentication is required and must be enabled. 0: Disable 1: Enable |
| udwEncryptEnabled | Enable or disable compatibility mode for encryption. This field should be disabled if compatibility is enabled (UNP1.0). 0: Disable 1: Enable |
| udwInterface | Optional. Required for IPC. Optional for NVR and VMS.  Copper network interface: 0  4G network interface: 1  Range: [0,1] |
| szLoginName | Optional. Required when AuthEnable is “Enable”. Username of the server. Chinese characters are not supported. Range: [0-63] |
| szPIN | Optional. The password range is [0, 256] when AuthEnable is enabled. Required when changing the password. |
| udwAddressType | AddressType: IP address type. 0: IPv4 1: IPv6 2: Domain name 3: Both IPv4 and IPv6 Currently only IPv4 is supported. |
| szAddress | Address |
| udwPort | Optional. Required for NVR and VMS. Corresponding server port of the compatibility mode. Range: [1-65535]. The default is 1701 for UNP1.0 and 5555 for UNP2.0. |
| udwUNPSuccessd | Indicates whether UNP connection is successful (whether the UNP is assigned IP, subnet mask, route). 0: Fail 1: Success  Required when using the GET method. |
| szUNPShareIPAddr | Assigned address when UNP is successfully connected to the server (returned only when query and connection succeed, not input parameter) |
| szUNPShareMask | Assigned subnet mask when UNP is successfully connected to the server (returned only when query and connection succeed, not input parameter) |
| szUNPShareWayAddr | Assigned route when UNP is successfully connected to the server (returned only when query and connection succeed, not input parameter) |
| byRes | Reserved field |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Device DNS info

|  |
| --- |
| typedef struct tagNETDEVDNSInfo  {  UINT32 udwNum;  [NETDEV\_DNS\_ADDRESS\_S](#_DNS地址) astDNSList[NETDEV\_DNS\_LIST\_NUM];  BYTE byRes[128];  }NETDEV\_DNS\_INFO\_S, \*LPNETDEV\_DNS\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of DNS addresses |
| astDNSList | DNS address list. The 1st is the preferred address, and the 2nd is the alternate address. |
| byRes | Reserved field |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### DNS address

|  |
| --- |
| typedef struct tagDNSAddress  {  UINT32 udwAddressType;  CHAR szIPAddress[NETDEV\_LEN\_64];  CHAR szIPv6Address[NETDEV\_LEN\_64];  BYTE byRes[128];  }NETDEV\_DNS\_ADDRESS\_S, \*LPNETDEV\_DNS\_ADDRESS\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwAddressType | IP address type. 0: IPv4 1: IPv6 3: Both IPv4 and IPv6 are required. Currently only IPv4 is supported |
| szIPAddress | IPv4 Address |
| szIPv6Address | IPv6 Address |
| byRes | Reserved field |

**See also:**

[NETDEV\_DNS\_INFO\_S](#_设备DNS信息)

### NIC info

|  |
| --- |
| typedef struct tagNETDEVNetworkCardsInfo  {  UINT32 udwNum;  UINT32 udwDefaultRouteNIC;  UINT32 udwWorkMode;  [NETDEV\_NETWORK\_INTERFACE\_INFO\_S](#_网卡的详细信息) astNetworkInterfaceList[NETDEV\_LEN\_8];  BYTE byRes [128];  }NETDEV\_NETWORK\_CARD\_INFO\_S, \*LPNETDEV\_NETWORK\_CARD\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of network interface cards (NICs) |
| udwDefaultRouteNIC | ID of the NIC used as the default route. Required when multiple NICs are configured (not including internal NIC). |
| udwWorkMode | NIC working mode. 0: Multi-address 1: Load balance 2: Network fault tolerance |
| astNetworkInterfaceList | NIC list. Optional when Num is 0. |
| byRes | Reserved field |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Detailed NIC info

|  |
| --- |
| typedef struct tagNETDEVNetworkInterfaceInfo  {  UINT32 udwID;  CHAR szName[NETDEV\_NAME\_MAX\_LEN];  UINT32 udwWorkMode;  BOOL bIsInnerNIC;  CHAR szInnerNICIPAddress[NETDEV\_IPADDR\_STR\_MAX\_LEN];  CHAR szInnerNICNetmask[NETDEV\_IPADDR\_STR\_MAX\_LEN];  CHAR szInnerNICName[NETDEV\_NAME\_MAX\_LEN];  UINT32 udwMTU;  CHAR szMAC[NETDEV\_NETWORK\_MACNAME\_LEN];  UINT32 udwNegotiationMode;  [NETDEV\_NETWORK\_IPV4\_INFO\_S](#_网卡信息中IPV4信息) stIPV4Infos;  [NETDEV\_NETWORK\_IPV6\_INFO\_S](#_网卡信息中IPV6信息) stIPV6Infos;  BYTE byRes[128];  }NETDEV\_NETWORK\_INTERFACE\_INFO\_S, \*LPNETDEV\_NETWORK\_INTERFACE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | NIC ID. For VMS and IPC it starts from 1. For NVR this field is the resource ID. |
| szName | NIC name. Applicable to NVR and IPC. |
| udwWorkMode | NIC working mode. 0: Multi-address 1: Load balance 2: Network fault tolerance |
| bIsInnerNIC | Whether is an internal NIC |
| szInnerNICIPAddress | IP of inner NIC |
| szInnerNICNetmask | Subnet mask of inner NIC |
| szInnerNICName | Name of inner NIC. Unconfigurable. For search only. |
| udwMTU | MTU length. Range: [576,1500] |
| szMAC | MAC address. Read-only. Length range: [0, 48] |
| udwNegotiationMode | Working mode of network interface |
| stIPV4Infos | IPv4 info. Optional if IPv4 is not supported. |
| stIPV6Infos | IPv6 info. Optional if IPv6 is not supported. |
| byRes | Reserved field |

**See also:**

[NETDEV\_NETWORK\_CARD\_INFO\_S](#_网卡信息)

### NIC IPv4 info

|  |
| --- |
| typedef struct tagNetWorkIpv4info  {  UINT32 udwIPGetType;  CHAR szLoginName[NETDEV\_LEN\_64];  CHAR szPIN[NETDEV\_LEN\_128];  UINT32 udwAddressNum;  [NETDEV\_IPV4\_ADDRESS\_INFO\_S](#_IPV4详细地址信息) astIPv4AddressInfo[NETDEV\_LEN\_8];  }NETDEV\_NETWORK\_IPV4\_INFO\_S, \*LPNETDEV\_NETWORK\_IPV4\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwIPGetType | IP obtainment mode. 0: Static IP 1: PPPOE 2: DHCP |
| szLoginName | PPPoE name |
| szPIN | PPPoE password (MD5 encrypted) |
| udwAddressNum | Number of IPs |
| astIPv4AddressInfo | IPv4 info. Optional if IPv4 is not supported. |

**See also:**

[NETDEV\_NETWORK\_INTERFACE\_INFO\_S](#_网卡的详细信息)

### Detailed IPv4 address info

|  |
| --- |
| typedef struct tagIpv4Addressinfo  {  CHAR szAddress[NETDEV\_LEN\_64];  CHAR szNetmask[NETDEV\_LEN\_64];  CHAR szGateway[NETDEV\_LEN\_64];  }NETDEV\_IPV4\_ADDRESS\_INFO\_S, \*LPNETDEV\_IPV4\_ADDRESS\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szAddress | IP address |
| szNetmask | Subnet mask |
| szGateway | Default gateway |

**See also:**

[NETDEV\_NETWORK\_IPV4\_INFO\_S](#_网卡信息中IPV4信息)

### NIC IPv6 info

|  |
| --- |
| typedef struct tagNetWorkIpv6info  {  UINT32 udwIPGetType;  UINT32 udwAddressNum;  [NETDEV\_IPV6\_ADDRESS\_INFO\_S](#_IPV6详细地址信息) astIPv6AddressInfo[NETDEV\_LEN\_8];  }NETDEV\_NETWORK\_IPV6\_INFO\_S, \*LPNETDEV\_NETWORK\_IPV6\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwIPGetType | IP obtainment mode |
| udwAddressNum | Number of IPs |
| astIPv6AddressInfo | IPv6 info. Optional if IPv6 is not supported. |

**See also:**

[NETDEV\_NETWORK\_INTERFACE\_INFO\_S](#_网卡的详细信息)

### Detailed IPv6 address info

|  |
| --- |
| typedef struct tagIpv6Addressinfo  {  UINT32 udwPrefixLenth;  CHAR szAddress[NETDEV\_LEN\_64];  CHAR szGateway[NETDEV\_LEN\_64];  }NETDEV\_IPV6\_ADDRESS\_INFO\_S, \*LPNETDEV\_IPV6\_ADDRESS\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPrefixLenth | Subnet prefix length: [3,127] |
| szAddress | IP address |
| szGateway | Default gateway |

**See also:**

[NETDEV\_NETWORK\_IPV6\_INFO\_S](#_网卡信息中IPV6信息)

### Port information

|  |
| --- |
| typedef struct tagNETDEVNetworkPortsInfo  {  UINT32 udwHttpPort;  UINT32 udwHttpsPort;  UINT32 udwRtspPort;  BYTE byRes[64];  }NETDEV\_NETWORK\_PORTS\_INFO\_S, \*LPNETDEV\_NETWORK\_PORTS\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwHttpPort | HTTP port |
| udwHttpsPort | HTTPS port |
| udwRtspPort | RTSP port |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Privacy mask configuration info

|  |
| --- |
| typedef struct tagPrivacyMaskPara  {  INT32 dwSize;  [NETDEV\_PRIVACY\_MASK\_AREA\_INFO\_S](#_区域配置结构体定义)  astArea[NETDEV\_MAX\_PRIVACY\_MASK\_AREA\_NUM];  }NETDEV\_PRIVACY\_MASK\_CFG\_S, \*LPNETDEV\_PRIVACY\_MASK\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Number of areas |
| astArea | Mask area parameters |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Structure of area configuration

|  |
| --- |
| typedef struct tagAreaInfo  {  INT32 bIsEanbled;  INT32 dwTopLeftX;  INT32 dwTopLeftY;  INT32 dwBottomRightX;  INT32 dwBottomRightY;  INT32 dwIndex;  }NETDEV\_PRIVACY\_MASK\_AREA\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bIsEanbled | Enable or disable |
| dwTopLeftX | X-coordinate of top left corner [0, 10000] |
| dwTopLeftY | Y-coordinate of top left corner [0, 10000] |
| dwBottomRightX | X-coordinate of bottom right corner [0, 10000] |
| dwBottomRightY | Y-coordinate of bottom right corner [0, 10000] |
| dwIndex | Index |

**See also:**

[NETDEV\_PRIVACY\_MASK\_CFG\_S](#_隐私遮盖配置信息)

### Tampering detection info

|  |
| --- |
| typedef struct tagNETDEVTamperAlarmInfo  {  INT32 dwSensitivity;  BYTE byRes[256];  }NETDEV\_TAMPER\_ALARM\_INFO\_S, \*LPNETDEV\_TAMPER\_ALARM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSensitivity | Sensitivity |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Motion detection analysis info

|  |
| --- |
| typedef struct tagNETDEVMotionAlarmInfo  {  INT32 dwSensitivity;  INT32 dwObjectSize;  INT32 dwHistory;  INT16 awScreenInfo[NETDEV\_SCREEN\_INFO\_ROW][NETDEV\_SCREEN\_INFO\_COLUMN];  BYTE byRes[64];  }NETDEV\_MOTION\_ALARM\_INFO\_S, \*LPNETDEV\_MOTION\_ALARM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSensitivity | Sensitivity |
| dwObjectSize | Object size |
| dwHistory | Duration |
| awScreenInfo | On-screen block info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Cross line detection analysis info

|  |
| --- |
| typedef struct tagNETDEVCrossLineAlarmInfo  {  BOOL bEnable;  [NETDEV\_CROSS\_LINE\_POINT\_INFO\_S](#_区域信息) stAreaInfo[4];  BYTE byRes[64];  }NETDEV\_CROSS\_LINE\_ALARM\_INFO\_S, \*LPNETDEV\_CROSS\_LINE\_ALARM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnable | Enable or disable |
| stAreaInfo | Area info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Area info

|  |
| --- |
| typedef struct tagNETDEVCrossLinePointInfo  {  BOOL bEnable;  INT32 dwSensitivity;  INT32 dwDirection;  INT32 dwTopLeftX;  INT32 dwTopLeftY;  INT32 dwBottomRightX;  INT32 dwBottomRightY;  }NETDEV\_CROSS\_LINE\_POINT\_INFO\_S, \*LPNETDEV\_CROSS\_LINE\_POINT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnable | Enable or disable |
| dwSensitivity | Sensitivity |
| dwDirection | Trigger direction |
| dwTopLeftX | x-coordinate of top left corner [0, 10000] |
| dwTopLeftY | y-coordinate of top left corner [0, 10000] |
| dwBottomRightX | x-coordinate of bottom right corner [0, 10000] |
| dwBottomRightY | y-coordinate of bottom right corner [0, 10000] |

**See also:**

[NETDEV\_CROSS\_LINE\_ALARM\_INFO\_S](#_越界检测分析信息)

### Intrusion detection info

|  |
| --- |
| typedef struct tagNETDEVIntrusionAlarmInfo  {  BOOL bEnable;  [NETDEV\_INTRUSION\_AREA\_INFO\_S](#_入侵检测区域信息) stAreaInfo[NETDEV\_LEN\_4];  BYTE byRes[64];  }NETDEV\_INTRUSION\_ALARM\_INFO\_S, \*LPNETDEV\_INTRUSION\_ALARM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnable | Enable or disable |
| stAreaInfo | Area information. Up to 4 areas are allowed. Each area consists of 3 to 6 sides. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Intrusion detection area info

|  |
| --- |
| typedef struct tagNETDEVIntrusionAreaInfo  {  BOOL bEnable;  INT32 dwSensitivity;  INT32 dwTimeThreshold;  INT32 percentage;  INT32 dwPointCount;  [NETDEV\_INTRUSION\_POINT\_INFO\_S](#_入侵检测区域点坐标) stPointInfo[NETDEV\_LEN\_6];  BYTE byRes[64];  }NETDEV\_INTRUSION\_AREA\_INFO\_S, \*LPNETDEV\_INTRUSION\_AREA\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnable | Enable or disable |
| dwSensitivity | Sensitivity. Range: 1-100 |
| dwTimeThreshold | Time threshold. Range: 1-10 |
| percentage | Percentage. Range: 1-100 |
| dwPointCount | Number of vertices. Range: 3-6. Ignored if no detection area is set. |
| stPointInfo | Coordinates of vertices |
| byRes | Reserved field. |

**See also:**

[NETDEV\_INTRUSION\_ALARM\_INFO\_S](#_入侵检测分析信息)

### Coordinates of intrusion detection area

|  |
| --- |
| typedef struct tagNETDEVIntrusionPointInfo  {  INT32 dwPointX;  INT32 dwPointY;  }NETDEV\_INTRUSION\_POINT\_INFO\_S, \*LPNETDEV\_INTRUSION\_POINT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwPointX | X [0, 10000] |
| dwPointY | Y [0, 10000] |

**See also:**

[NETDEV\_INTRUSION\_AREA\_INFO\_S](#_入侵检测区域信息)

### HDD list

|  |
| --- |
| typedef struct tagNETDEVDiskInfoList  {  INT32 dwSize;  [NETDEV\_DISK\_INFO\_S](#_硬盘详细信息) astDisksInfo[NETDEV\_DISK\_MAX\_NUM];  }NETDEV\_DISK\_INFO\_LIST\_S,\*LPNETDEV\_DISK\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Number of HDDs |
| astDisksInfo | HDD info |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Detailed HDD info

|  |
| --- |
| typedef struct tagNETDEVDiskInfo  {  INT32 dwDiskCabinetIndex;  INT32 dwSlotIndex;  INT32 dwTotalCapacity;  INT32 dwUsedCapacity;  [NETDEV\_DISK\_WORK\_STATUS\_E](#_磁盘工作状态枚举) enStatus;  CHAR szManufacturer[NETDEV\_LEN\_32];  }NETDEV\_DISK\_INFO\_S,\*LPNETDEV\_DISK\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwDiskCabinetIndex | HDD index |
| dwSlotIndex | HDD slot index |
| dwTotalCapacity | Total HDD capacity |
| dwUsedCapacity | Used capacity |
| enStatus | Operation status |
| szManufacturer | Manufacturer |

**See also:**

[NETDEV\_DISK\_INFO\_LIST\_S](#_硬盘信息列表)

### Focus info

|  |
| --- |
| typedef struct tagNETDEVFocusInfo  {  [NETDEV\_FOCUS\_MODE\_E](#_聚焦模式) enFocusMode;  BYTE bRes[16];  }NETDEV\_FOCUS\_INFO\_S, \*LPNETDEV\_FOCUS\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| enFocusMode | Focus mode |
| bRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Day/night mode info

|  |
| --- |
| typedef struct tagNETDEVIrFilterInfo  {  [NETDEV\_IR\_CUT\_FILTER\_MODE\_E](#_昼夜模式) enIrCutFilterMode;  BYTE bRes[16];  }NETDEV\_IRCUT\_FILTER\_INFO\_S, \*LPNETDEV\_IRCUT\_FILTER\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| enIrCutFilterMode | Day/night mode |
| bRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Defog info

|  |
| --- |
| typedef struct tagNETDEVDefoggingInfo  {  [NETDEV\_DEFOGGING\_MODE\_E](#_透雾模式枚举) enDefoggingMode;  FLOAT fDefoggingLevel;  BYTE bRes[64];  }NETDEV\_DEFOGGING\_INFO\_S, \*LPNETDEV\_DEFOGGING\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| enDefoggingMode | Defog mode |
| fDefoggingLevel | Defog level (0.0, 1.0) |
| bRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Image enhancement parameters of specified channel

|  |
| --- |
| typedef struct tagNETDEVImagingEnhance  {  UINT32 udwBrightness;  UINT32 udwContrast;  UINT32 udwSaturation;  UINT32 udwSharpness;  UINT32 udwImageRotation;  UINT32 udw2DNoiseReduce;  UINT32 udw3DNoiseReduce;  BYTE byRes[252];  }NETDEV\_IMAGE\_ENHANCE\_S, \*LPNETDEV\_IMAGE\_ENHANCE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwBrightness | Brightness |
| udwContrast | Contrast |
| udwSaturation | Saturation |
| udwSharpness | Sharpness |
| udwImageRotation | Image rotation modes |
| udw2DNoiseReduce | 2D noise reduction |
| udw3DNoiseReduce | 3D noise reduction |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Info about audio input parameters

|  |
| --- |
| typedef struct tagNETDEVAudioInputCfgInfo  {  BOOL bIsMute;  UINT32 udwType;  UINT32 udwEncodeFormat;  UINT32 udwSampleRate;  UINT32 udwInputGain;  BOOL bNoiseReductionEnabled;  INT32 dwAudioInNum;  [NETDEV\_AUDIO\_INPUT\_INFO\_S](#_音频输入通道信息) astAudioInputInfoList[NETDEV\_AUDIO\_IN\_MAX\_NUM];  INT32 dwSerialInNum;  [NETDEV\_AUDIO\_INPUT\_INFO\_S](#_音频输入通道信息) astSerialInputInfoList[NETDEV\_SERIAL\_IN\_MAX\_NUM];  BYTE byRes[256];  }NETDEV\_AUDIO\_INPUT\_CFG\_INFO\_S, \*LPNETDEV\_AUDIO\_INPUT\_CFG\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bIsMute | Whether sound is mute. 0: No 1: Yes |
| udwType | Collection port type. See [NETDEV\_AUDIO\_COLLECTION\_PORT\_TYPE\_E](#_音频采集口枚举). |
| udwEncodeFormat | Audio encoding format. See [NETDEV\_AUDIO\_ENCODE\_FORMAT\_TYPE\_E](#_音频编码格式类型枚举). |
| udwSampleRate | Audio sampling rate. See [NETDEV\_AUDIO\_SAMPLING\_RATE\_E](#_音频采样率枚举). |
| udwInputGain | Audio gain. Range: [0,255] |
| bNoiseReductionEnabled | Enable or disable noise reduction. 0: Disable 1: Enable |
| dwAudioInNum | Number of audio inputs |
| astAudioInputInfoList | Audio input list. Valid when the type of audio collection port is [NETDEV\_AUDIO\_COLLECTION\_PORT\_TYPE\_AUDIO\_IN](#_音频采集口枚举). |
| dwSerialInNum | Number of serial inputs |
| astSerialInputInfoList | Serial input list |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Audio input channel info

|  |
| --- |
| typedef struct tagNETDEVAudioInputInfo  {  INT32 dwChannelID;  UINT32 udwEnabled;  UINT32 udwMode;  BYTE byRes[128];  }NETDEV\_AUDIO\_INPUT\_INFO\_S, \*LPNETDEV\_AUDIO\_INPUT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Audio channel ID |
| udwEnabled | Whether audio input channel is enabled |
| udwMode | Audio input channel mode. See [NETDEV\_AUDIO\_CHL\_MODE\_E](#_音频输入模式枚举). |
| byRes | Reserved field |

**See also:**

[NETDEV\_AUDIO\_INPUT\_CFG\_INFO\_S](#_音频输入参数配置信息)

### DST configuration info

|  |
| --- |
| typedef struct tagNETDEVDSTCfgInfo  {  BOOL bEnableDST;  [NETDEV\_TIME\_DST\_CFG\_S](#_时间配置) stTimeDSTCfg;  BYTE byRes[128];  }NETDEV\_DST\_CFG\_S, \*LPNETDEV\_DST\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnableDST | Whether DST is enabled |
| stTimeDSTCfg | DST configuration |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### DST configuration

|  |
| --- |
| typedef struct tagNETDEVSystemTimeDSTCfg  {  [NETDEV\_TIME\_DST\_S](#_时间配置_1) stBeginTime;  [NETDEV\_TIME\_DST\_S](#_时间配置_1) stEndTime;  INT32 dwOffsetTime;  }NETDEV\_TIME\_DST\_CFG\_S, \*LPNETDEV\_TIME\_DST\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stBeginTime | DST begin time |
| stEndTime | DST end time |
| dwOffsetTime | DST offset time. See [NETDEV\_DST\_OFFSET\_TIME](#_夏令时偏移时间). |

**See also:**

[NETDEV\_DST\_CFG\_S](#_夏令时配置信息)

### Time configuration

|  |
| --- |
| typedef struct tagNETDEVSystemTimeDST  {  INT32 dwMonth;  INT32 dwWeekInMonth;  INT32 dwDayInWeek;  INT32 dwHour;  }NETDEV\_TIME\_DST\_S, \*LPNETDEV\_TIME\_DST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwMonth | Month (1 to 12) |
| dwWeekInMonth | Which week of the month (1 to 5) |
| dwDayInWeek | What day in the week. See [NETDEV\_DAY\_IN\_WEEK\_E](#_星期). |
| dwHour | Hour |

**See also:**

[NETDEV\_TIME\_DST\_CFG\_S](#_时间配置)

### Recording schedule configuration info

|  |
| --- |
| typedef struct tagNETDEVRecordPlanCfgInfo  {  BOOL bPlanEnable;  BOOL bRedundantStorage;  [NETDEV\_RECORD\_RULE\_S](#_录像计划规则) stRecordRule;  [NETDEV\_VIDEO\_WEEK\_PLAN\_S](#_视频计划（周）配置) stWeekPlan;  UINT32 udwChlID;  UINT32 udwReqSeq;  UINT32 udwTamplateID;  CHAR szTamplateName[NETDEV\_NAME\_MAX\_LEN];  BYTE byRes[116];  }NETDEV\_RECORD\_PLAN\_CFG\_INFO\_S, \*LPNETDEV\_RECORD\_PLAN\_CFG\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bPlanEnable | Whether schedule is enabled |
| bRedundantStorage | Whether redundant recording is enabled. Applicable to NVR only. |
| stRecordRule | Recording schedule rules. Applicable to NVR only. |
| stWeekPlan | Schedule configuration. Applicable to NVR only. |
| udwChlID | Video input channel ID. Used when adding or getting in batches. |
| udwReqSeq | Sequence number of request data. Range: [1, 50]. Applicable to VMS only. Required when adding a recording schedule. |
| udwTamplateID | Time template ID |
| szTamplateName | Time template name. Applicable to VMS only. Returned by the GET interface. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Recording schedule rules

|  |
| --- |
| typedef struct tagNETDEVRecordRule  {  UINT32 udwPreRecordTime;  UINT32 udwPostRecordTime;  BYTE byRes[32];  }NETDEV\_RECORD\_RULE\_S, \*LPNETDEV\_RECORD\_RULE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPreRecordTime | Pre-alarm recording time. See [NETDEV\_PRE\_RECORD\_TIME\_E](#_警前预录时间枚举). |
| udwPostRecordTime | Post-alarm recording time. See [NETDEV\_POST\_RECORD\_TIME\_E](#_警后预录时间枚举定义). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_RECORD\_PLAN\_CFG\_INFO\_S](#_录像计划配置信息)

### Recording schedule (weekly) configuration

|  |
| --- |
| typedef struct tagNETDEVVideoPlanWeek  {  BOOL bEnabled;  UINT32 udwDayNum;  [NETDEV\_VIDEO\_DAY\_PLAN\_S](#_视频计划（天）配置) astDayPlan[NETDEV\_MAX\_DAY\_NUM];  BYTE byRes[32];  }NETDEV\_VIDEO\_WEEK\_PLAN\_S, \*LPNETDEV\_VIDEO\_WEEK\_PLAN\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Whether arming schedule is enabled. Only IPC supports the enable/disable switch; for VMS and NVR arming schedule is enabled by default. See [NETDEV\_DEFENCE\_PLAN\_ENABLE\_E](#_布防计划使能枚举). |
| udwDayNum | Number of scheduled days.  NVR: Max 8 (7 days of a week plus holiday). IPC: Max 7 (7 days of a week). |
| astDayPlan | Arming schedule list of every day in a week. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_RECORD\_PLAN\_CFG\_INFO\_S](#_录像计划配置信息)

### Recording schedule (daily) configuration

|  |
| --- |
| typedef struct tagNETDEVVideoDayPlan  {  UINT32 udwIndex;  UINT32 udwSectionNum;  [NETDEV\_VIDEO\_TIME\_SECTION\_S](#_视频时间段配置) astTimeSection[NETDEV\_MAX\_TIME\_SECTION\_NUM];  }NETDEV\_VIDEO\_DAY\_PLAN\_S, \*LPNETDEV\_VIDEO\_DAY\_PLAN\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwIndex | Index of days in a week. See [NETDEV\_LAPI\_WEEK\_INFO\_E](#_星期枚举). |
| udwSectionNum | Number of time periods per day. NVR: Max 8; IPC: Max 4. |
| astTimeSection | Arming period configuration |

**See also:**

[NETDEV\_VIDEO\_WEEK\_PLAN\_S](#_视频计划（周）配置)

### Video time period configuration

|  |
| --- |
| typedef struct tagNETDEVVideoTimeSection  {  CHAR szBeginTime[NETDEV\_LEN\_64];  CHAR szEndTime[NETDEV\_LEN\_64];  UINT32 udArmingType;  }NETDEV\_VIDEO\_TIME\_SECTION\_S, \*LPNETDEV\_VIDEO\_TIME\_SECTION\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szBeginTime | Start time, for example, "00:00:00" |
| szEndTime | End time, for example, "24:00:00" |
| udArmingType | Arming type. See [NETDEV\_ARMING\_TYPE\_E](#_告警布防类型枚举). |

**See also:**

[NETDEV\_VIDEO\_DAY\_PLAN\_S](#_视频计划（天）配置)

### Snapshot info of video input channel

|  |
| --- |
| typedef struct tagNETDEVVideoSnapshotInfo  {  BOOL bIsEnabled;  UINT32 udwPictureMaxSize;  UINT32 udwSnapshotInterval;  UINT32 udwSnapshotNum;  [NETDEV\_VIDEO\_RESOLUTION\_S](#_视频源分辨率信息) stResolution;  BYTE byRes[255];  }NETDEV\_VIDEO\_SNAPSHOT\_S,\*LPNETDEV\_VIDEO\_SNAPSHOT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bIsEnabled | Enable or disable snapshot. 0: Disable 1: Enable |
| udwPictureMaxSize | Maximum image size |
| udwSnapshotInterval | Snapshot interval (unit: second). Range: [1, 60] |
| udwSnapshotNum | Number of snapshots. Range: [1,3] |
| stResolution | Snapshot resolution |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Video source resolution information

|  |
| --- |
| typedef struct tagNETDEVVideoResolution  {  INT32 dwWidth;  INT32 dwHeight;  }NETDEV\_VIDEO\_RESOLUTION\_S, \*LPNETDEV\_VIDEO\_RESOLUTION\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwWidth | Width |
| dwHeight | Height |

**See also:**

[NETDEV\_VIDEO\_SNAPSHOT\_S](#_视频输入通道的抓图信息)

### Schedule (weekly) configuration

|  |
| --- |
| typedef struct tagNETDEVPlanWeekInfo  {  BOOL bEnabled;  UINT32 udwNum;  [NETDEV\_DAY\_PLAN\_INFO\_S](#_计划（天）配置) astDayPlanInfo[NETDEV\_MAX\_DAY\_NUM];  BYTE byRes[512];  }NETDEV\_WEEK\_PLAN\_INFO\_S, \*LPNETDEV\_WEEK\_PLAN\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Whether schedule is enabled. Applicable to IPC only. |
| udwNum | Number of scheduled days. NVR: 7 days of a week plus holiday. IPC: 7 days of a week. |
| astDayPlanInfo | Arming schedule list of every day in a week. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Schedule (daily) configuration

|  |
| --- |
| typedef struct tagNETDEVDayPlanInfo  {  UINT32 udwID;  UINT32 udwNum;  [NETDEV\_TIME\_SECTION\_INFO\_S](#_时间段配置) astTimeSection[NETDEV\_MAX\_TIME\_SECTION\_NUM];  }NETDEV\_DAY\_PLAN\_INFO\_S, \*LPNETDEV\_DAY\_PLAN\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Index of days in a week. 1: Monday 2: Tuesday 3: Wednesday 4: Thursday 5: Friday 6: Saturday 7: Sunday 8: Holiday |
| udwNum | Number of time periods per day. NVR: Max 8 IPC: Max 4. |
| astTimeSection | Time period configuration. Time periods on the same day must not overlap. |

**See also:**

[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置)

### Time period configuration

|  |
| --- |
| typedef struct tagNETDEVTimeSectionInfo  {  CHAR szBeginTime[NETDEV\_LEN\_32];  CHAR szEndTime[NETDEV\_LEN\_32];  UINT32 udwArmingType;  }NETDEV\_TIME\_SECTION\_INFO\_S, \*LPNETDEV\_TIME\_SECTION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szBeginTime | Start time, for example, "00:00:00" |
| szEndTime | End time, for example, "24:00:00" |
| udwArmingType | Arming type: 0: Schedule 1: Motion detection 2: Alarm 3: Motion detection and alarm 4: Motion detection or alarm 5: No schedule 10: Event |

**See also:**

[NETDEV\_DAY\_PLAN\_INFO\_S](#_计划（天）配置)

### Arming schedule configuration of alarm input/output (weekly)

|  |
| --- |
| typedef struct tagNETDEVSwitchWeekPlanInfo  {  UINT32 udwSwitchIndex;  [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) astWeekPlanInfo;  BYTE byRes[128];  }NETDEV\_SWITCH\_WEEK\_PLAN\_INFO\_S, \*LPNETDEV\_SWITCH\_WEEK\_PLAN\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSwitchIndex | Alarm output index |
| astWeekPlanInfo | Weekly schedule configuration info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Recording status list

|  |
| --- |
| typedef struct tagNETDEVRecordStatusList  {  UINT32 udwSize;  [NETDEV\_RECORD\_STATUS](#_录像状态信息) astRecordStatus[NETDEV\_CHANNEL\_MAX];  }NETDEV\_RECORD\_STATUS\_LIST\_S, \*LPNETDEV\_RECORD\_STATUS\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSize | Number of recording statuses |
| astRecordStatus | Recording status info |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Recording status info

|  |
| --- |
| typedef struct tagNETDEVRecordStatus  {  INT32 dwChannelID;  INT32 dwRecordType;  INT32 dwRecordStatus;  BYTE byRes[128];  }NETDEV\_RECORD\_STATUS, \*LPNETDEV\_RECORD\_STATUS; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| dwRecordType | Recording type. 0: Manual recording 1: Event recording 2: POS recording 3: Schedule recording 4: Others |
| dwRecordStatus | Recording Status. 0: Recording in progress 1: Recording not started 2: No HDD or faulty HDD 3: Channel offline |
| byRes | Reserved field |

**See also:**

[NETDEV\_RECORD\_STATUS\_LIST\_S](#_录像状态信息列表)

### Serial port info

|  |
| --- |
| typedef struct tagNETDEVSerialPorts  {  INT32 dwSerialPortNum;  [LPNETDEV\_SERIAL\_PORT\_S](#_串口配置信息) pstSerialPortList;  BYTE byRes[512];  }NETDEV\_SERIAL\_PORTS\_S, \*LPNETDEV\_SERIAL\_PORTS\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSerialPortNum | Number of serial ports |
| pstSerialPortList | Serial port list. Optional when dwSerialPortNum is 0. Memory needs to be allocated dynamically when dwSerialPortNum is not 0. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Serial port configuration info

|  |
| --- |
| typedef struct tagNETDEVSerialPort  {  INT32 dwChannelID;  INT32 dwMode;  [NETDEV\_SERIAL\_PARAM\_S](#_串口参数) stSerialParam;  BYTE byRes[512];  }NETDEV\_SERIAL\_PORT\_S, \*LPNETDEV\_SERIAL\_PORT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Serial port number |
| dwMode | Working mode. 1: PTZ control 2: Transparent channel |
| stSerialParam | Serial port parameters |
| byRes | Reserved field |

**See also:**

[NETDEV\_SERIAL\_PORTS\_S](#_串口信息)

### Serial port parameters

|  |
| --- |
| typedef struct tagNETDEVSeriaParam  {  INT32 dwBaudRate;  INT32 dwDataBit;  INT32 dwFlowCtrl;  INT32 dwParity;  INT32 dwStopBit;  BYTE byRes[512];  }NETDEV\_SERIAL\_PARAM\_S, \*LPNETDEV\_SERIAL\_PARAM\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwBaudRate | Baud rate of serial port. Special value in the range of [1200,115200]. Unit: bps |
| dwDataBit | Data bit. Range: [5 -8] |
| dwFlowCtrl | Flow control. 0: None 1: By software 2: By hardware |
| dwParity | Parity bit. 0: None 1: Odd 2: Even |
| dwStopBit | Stop bit |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SERIAL\_PORT\_S](#_串口配置信息)

### Audio status list

|  |
| --- |
| typedef struct tagNETDEVAudioDecodeStatusList  {  INT32 dwSize;  [NETDEV\_AUDIO\_DECODE\_STATUS\_S](#_随路音频状态信息) astAudioDecStatus[NETDEV\_STREAM\_MAX\_NUM];  }NETDEV\_AUDIO\_DECODE\_STATUS\_LIST\_S, \*LPNETDEV\_AUDIO\_DECODE\_STATUS\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Quantity |
| astAudioDecStatus | Audio status info |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Audio status info

|  |
| --- |
| typedef struct tagNETDEVAudioDecodeStatus  {  INT32 dwStreamID;  BOOL bDecodeAudio;  }NETDEV\_AUDIO\_DECODE\_STATUS\_S, \*LPNETDEV\_AUDIO\_DECODE\_STATUS\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwStreamID | Video stream index number |
| bDecodeAudio | Enable or disable audio decoding |

**See also:**

[NETDEV\_AUDIO\_DECODE\_STATUS\_LIST\_S](#_随路音频状态信息列表)

### Video loss alarm configuration info

|  |
| --- |
| typedef struct tagNETDEVVideoLossRuleInfo  {  BOOL bEnabled;  BYTE byRes[256];  }NETDEV\_VIDEO\_LOSS\_RULE\_INFO\_S, \*LPNETDEV\_VIDEO\_LOSS\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Whether is enabled. 0: No 1: Yes |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Linkage action list

|  |
| --- |
| typedef struct tagNETDEVLinkageActionList  {  UINT32 udwNum;  [NETDEV\_LINKAGE\_ACTION\_INFO\_S](#_布控任务联动动作) stActionInfo[NETDEV\_MAX\_LINK\_ACTION\_NUM];  BYTE byRes[256];  }NETDEV\_LINKAGE\_ACTION\_LIST\_S,\*LPNETDEV\_LINKAGE\_ACTION\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of linkage actions |
| stActionInfo | Linkage actions |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Linkage actions of task

|  |
| --- |
| typedef struct tagNETDEVLinkageActionInfo  {  UINT32 udwActID;  [NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S](#_使能联动参数) stEnabledInfo;  [NETDEV\_OUTPUT\_SWITCH\_ACT\_PARAM\_INFO\_S](#_联动开关量输出) stOutputSwitchActParamInfo;  [NETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S](#_通道联动) stChannelActParamInfo;  [NETDEV\_PRESET\_ACT\_PARAM\_INFO\_S](#_联动云台预置位) stPresetActParamInfo;  BYTE byRes[512];  }NETDEV\_LINKAGE\_ACTION\_INFO\_S,\*LPNETDEV\_LINKAGE\_ACTION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwActID | Action ID. See [NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举). |
| stEnabledInfo | Enable or disable linkage action parameters. Applicable to alarm-triggered buzzer, email and pop-up window. |
| stOutputSwitchActParamInfo | Alarm output |
| stChannelActParamInfo | Trigger NVR preview, recording, and snapshot |
| stPresetActParamInfo | PTZ preset |
| byRes | Reserved field. |

**See also:**

[NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表)

### Channel linkage info

|  |
| --- |
| typedef struct tagNETDEVChannelActParamInfo  {  UINT32 udwNum;  INT32 adwChannelID[NETDEV\_CHANNEL\_MAX];  }NETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S,\*LPNETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of channels |
| adwChannelID | Channel ID list |

**See also:**

[NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举)

### Linkage action enablement parameter

|  |
| --- |
| typedef struct tagNETDEVEnabledActParamInfo  {  BOOL bEnabled;  BYTE byRes[64];  }NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S, \*LPNETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Enablement flag. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举)

### PTZ preset

|  |
| --- |
| typedef struct tagNETDEVPresetActParamInfo  {  UINT32 udwNum;  [NETDEV\_CHANNEL\_PRESET\_S](#_联动云台预置位信息) stChannelPreset[NETDEV\_CHANNEL\_MAX];  }NETDEV\_PRESET\_ACT\_PARAM\_INFO\_S,\*LPNETDEV\_PRESET\_ACT\_PARAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of linkage actions |
| stChannelPreset | Linked PTZ preset list |

**See also:**

[NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举)

### Info about linked PTZ preset

|  |
| --- |
| typedef struct tagNETDEVChannelPreset  {  INT32 dwChannelID;  INT32 dwPresetID;  BYTE byRes[128];  }NETDEV\_CHANNEL\_PRESET\_S,\*LPNETDEV\_CHANNEL\_PRESET\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| dwPresetID | Preset ID |
| byRes | Reserved field. |

**See also:**

[NETDEV\_PRESET\_ACT\_PARAM\_INFO\_S](#_联动云台预置位)

### Alarm output

|  |
| --- |
| typedef struct tagNETDEVOutputSwitchActParamInfo  {  UINT32 udwNum;  [NETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S](#_输出开关量的逻辑报警状态(手动告警)_2)  astOutputAlarmStatusInfo[NETDEV\_MAX\_ALARM\_OUT\_NUM];  }NETDEV\_OUTPUT\_SWITCH\_ACT\_PARAM\_INFO\_S,\*LPNETDEV\_OUTPUT\_SWITCH\_ACT\_PARAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of linked alarm outputs |
| astOutputAlarmStatusInfo | List of linked alarm outputs |

**See also:**

[NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举)

### Logical alarm status of alarm output (manual alarm)

|  |
| --- |
| typedef struct tagNETDEVOutputSwitchesAlarmStatus  {  INT32 dwBooleanId;  INT32 dwChannelId;  INT32 enAlarmStatus;  } NETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S, \*LPNETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwBooleanId | Alarm output ID |
| dwChannelId | Channel ID. For the device it is 0. |
| enAlarmStatus | Alarm output status. See [NETDEV\_RELAYOUTPUT\_STATE\_E](#_开关量状态枚举). |

**See also:**

[NETDEV\_OUTPUT\_SWITCH\_ACT\_PARAM\_INFO\_S](#_联动开关量输出结构体)

### Linkage action enablement parameters

|  |
| --- |
| typedef struct tagNETDEVEnabledActParamInfo  {  BOOL bEnabled;  BYTE byRes[64];  }NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S, \*LPNETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Enablement flag. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举)

### Tampering detection configuration info

|  |
| --- |
| typedef struct tagNETDEVTamperDetectionRuleInfo  {  BOOL bEnabled;  UINT32 udwSensitivity;  UINT32 udwDuration;  BYTE byRes[256];  }NETDEV\_TAMPER\_DETECTION\_RULE\_INFO\_S, \*LPNETDEV\_TAMPER\_DETECTION\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Whether is enabled. 0: No 1: Yes |
| udwSensitivity | Sensitivity. Range: [1,100] |
| udwDuration | Duration (unit: second) Range: [0, 10]. Required for IPC. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Structure of info about temperature detection blackbody

|  |
| --- |
| typedef struct tagNETDEVTempDetectionBlackBodyInfo  {  FLOAT fTemperature;  [NETDEV\_XW\_AREA\_S](#_相对于电视墙的位置信息) stAreaInfo;  BYTE byRes[512];  }NETDEV\_TEMP\_DETECTION\_BLACKBODY\_INFO\_S,\*LPNETDEV\_TEMP\_DETECTION\_BLACKBODY\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| fTemperature | Blackbody temperature (unit: ℃, 2 decimal places). Range: -40.00 to 150.00 |
| stAreaInfo | Rectangle area info. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Structure of info about temperature detection correction

|  |
| --- |
| typedef struct tagNETDEVTempDetectionCorrectInfo  {  INT32 dwMode;  FLOAT fCorrectionTemperature;  FLOAT fEnvironmentTemperature;  BYTE byRes[512];  }NETDEV\_TEMP\_DETECTION\_CORRECT\_INFO\_S,\*LPNETDEV\_TEMP\_DETECTION\_CORRECT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwMode | Correction mode. 0: Inside body 1: Body surface |
| fCorrectionTemperature | Correction temperature (unit: ℃, 2 decimal places). Range: -10.00 to 10.00 |
| fEnvironmentTemperature | Environment temperature (unit: ℃, 2 decimal places). Range: -40.00 to 100.00 |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Structure of info about temperature detection unit

|  |
| --- |
| typedef struct tagNETDEVTempDetectioUnitInfo  {  INT32 dwUnit;  BYTE byRes[512];  }NETDEV\_TEMP\_DETECTION\_UNIT\_INFO\_S,\*LPNETDEV\_TEMP\_DETECTION\_UNIT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwUnit | Temperature unit. 0: °C 1: °F 2: K |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Motion detection area type info

|  |
| --- |
| typedef struct tagNETDEVMotionDetectionAreaType  {  UINT32 udwAreaType;  BYTE byRes[128];  }NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_S, \*LPNETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwAreaType | Area type. See [NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_E](#_运动检测区域类型枚举). |
| byRes | Reserved field |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Motion detection grid area info

|  |
| --- |
| typedef struct tagNETDEVMotionDetectionAreaGridInfo  {  BOOL bEnabled;  UINT32 udwSensitivity;  INT16 awGridInfo[NETDEV\_SCREEN\_INFO\_ROW][NETDEV\_SCREEN\_INFO\_COLUMN];  BYTE byRes[256];  }NETDEV\_MOTION\_DETECTION\_AREA\_GRID\_INFO\_S, \*LPNETDEV\_MOTION\_DETECTION\_AREA\_GRID\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Whether detection area is enabled. |
| udwSensitivity | Motion detection sensitivity. The greater the value, the higher the sensitivity. Range: [1,100] |
| awGridInfo | Motion detection grid info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Motion detection rectangle area list

|  |
| --- |
| typedef struct tagNETDEVMotionDetectionAreaRectangleInfoList  {  UINT32 udwNum;  [NETDEV\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_S](#_运动检测矩形区域信息) astRectangleAreaInfoList[NETDEV\_LEN\_8];  BYTE byRes[128];  }NETDEV\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_LIST\_S, \*LPNETDEV\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of rectangle areas |
| astRectangleAreaInfoList | Rectangle area list |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Motion detection rectangle area info

|  |
| --- |
| typedef struct tagNETDEVMotionDetectionAreaRectangleInfo  {  UINT32 udwID;  BOOL bEnabled;  UINT32 udwSensitivity;  UINT32 udwDuration;  UINT32 udwTargetSize;  [NETDEV\_RECT\_AREA\_INFO\_S](#_矩形区域坐标信息) stRectAreaInfo;  BYTE byRes[128];  }NETDEV\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_S, \*LPNETDEV\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Rectangle area ID |
| bEnabled | Whether detection area is enabled |
| udwSensitivity | Motion detection sensitivity. The greater the value, the higher the sensitivity. Range: [1,100] |
| udwDuration | Duration. Range: [1,100] |
| udwTargetSize | Object size. Range: [1,100] |
| stRectAreaInfo | Rectangle area info. Coordinates range: [0,9999] |
| byRes | Reserved field. |

**See also:**

[NETDEV\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_LIST\_S](#_运动检测矩形区域信息列表)

### Coordinates info of rectangle area

|  |
| --- |
| typedef struct tagNETDEVRectAreaInfo  {  UINT32 udwTopLeftX;  UINT32 udwTopLeftY;  UINT32 udwBottomRightX;  UINT32 udwBottomRightY;  }NETDEV\_RECT\_AREA\_INFO\_S, \*LPNETDEV\_RECT\_AREA\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTopLeftX | x-coordinate of top left |
| udwTopLeftY | y-coordinate of top left |
| udwBottomRightX | x-coordinate of bottom right |
| udwBottomRightY | y-coordinate of bottom right |

**See also:**

[NETDEV\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_S](#_运动检测矩形区域信息)

### Linkage actions of alarm input/output

|  |
| --- |
| typedef struct tagNETDEVSwitchLinkageAction  {  UINT32 udwSwitchIndex;  [NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表) stLinkageActionList;  BYTE byRes[256];  }NETDEV\_SWITCH\_LINKAGE\_ACTION\_S,\*LPNETDEV\_SWITCH\_LINKAGE\_ACTION\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSwitchIndex | Alarm output index |
| stLinkageActionList | Task linkage actions list |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### RAID status info

|  |
| --- |
| typedef struct tagNETDEVRAIDStatus  {  BOOL bEnabled;  BYTE byRes[128];  }NETDEV\_RAID\_STATUS\_S, \*LPNETDEV\_RAID\_STATUS\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Enable or disable RAID. 0: Disable 1: Enable |
| byRes | Reserved field |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### HDD list

|  |
| --- |
| typedef struct tagNETDEVHDDInfoList  {  INT32 dwSize;  [NETDEV\_HDD\_INFO\_S](#_硬盘信息) astHDDInfo[NETDEV\_LEN\_32];  BYTE byRes[128];  }NETDEV\_HDD\_INFO\_LIST\_S,\*LPNETDEV\_HDD\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Number of HDDs |
| astHDDInfo | HDD info |
| byRes | Reserved field |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### HDD info

|  |
| --- |
| typedef struct tagNETDEVHDDInfo  {  UINT32 udwID;  UINT32 udwType;  UINT32 udwWorkMode;  UINT32 udwTotalCapacity;  CHAR szRAIDName[NETDEV\_LEN\_64];  UINT32 udwStatus;  CHAR szManufacturer[NETDEV\_LEN\_64];  BYTE byRes[128];  }NETDEV\_HDD\_INFO\_S,\*LPNETDEV\_HDD\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | HDD ID |
| udwType | HDD type. See [NETDEV\_HDD\_TYPE\_E](#_磁盘类型). |
| udwWorkMode | Working mode of HDD. See [NETDEV\_HDD\_WORK\_MODE\_E](#_磁盘工作模式) |
| udwTotalCapacity | Total capacity of HDD (MB) |
| szRAIDName | RAID name |
| udwStatus | HDD status. See [NETDEV\_HDD\_STATUS\_E](#_磁盘状态). |
| szManufacturer | Manufacturer name |
| byRes | Reserved field |

**See also:**

[NETDEV\_HDD\_INFO\_LIST\_S](#_磁盘信息列表)

### Storage container list

|  |
| --- |
| typedef struct tagNETDEVStorageContainerInfoList  {  UINT32 udwLocalHDDNum;  [NETDEV\_STORAGE\_CONTAINER\_INFO\_S](#_存储容器信息) astLocalHDDList[NETDEV\_LOCAL\_DISK\_MAX\_NUM];  UINT32 udwSDNum;  [NETDEV\_STORAGE\_CONTAINER\_INFO\_S](#_存储容器信息) astSDList[NETDEV\_SD\_CARD\_DISK\_MAX\_NUM];  UINT32 udwArrayNum;  [NETDEV\_STORAGE\_CONTAINER\_INFO\_S](#_存储容器信息) astArrayList[NETDEV\_ARRAY\_MAX\_NUM];  UINT32 udwExtendCabinet1HDDNum;  [NETDEV\_STORAGE\_CONTAINER\_INFO\_S](#_存储容器信息) astExtendCabinet1HDDList[NETDEV\_EXTEND\_CABINET\_DISK\_MAX\_NUM];  UINT32 udwExtendCabinet2HDDNum;  [NETDEV\_STORAGE\_CONTAINER\_INFO\_S](#_存储容器信息) astExtendCabinet2HDDList[NETDEV\_EXTEND\_CABINET\_DISK\_MAX\_NUM];  UINT32 udwNASNum;  [NETDEV\_EXTEND\_STORAGE\_CONTAINER\_INFO\_S](#_扩展存储容器信息) astNASList[NETDEV\_NAS\_MAX\_NUM];  UINT32 udweSATANum;  [NETDEV\_EXTEND\_STORAGE\_CONTAINER\_INFO\_S](#_扩展存储容器信息) asteSATAList[NETDEV\_ESATA\_MAX\_NUM];  UINT32 udwIPSANNum;  [NETDEV\_EXTEND\_STORAGE\_CONTAINER\_INFO\_S](#_扩展存储容器信息) astIPSANList[NETDEV\_IPSAN\_MAX\_NUM];  BYTE byRes[256];  }NETDEV\_STORAGE\_CONTAINER\_INFO\_LIST\_S,\*LPNETDEV\_STORAGE\_CONTAINER\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwLocalHDDNum | Number of local HDDs |
| astLocalHDDList | Local HDD list |
| udwSDNum | Number of SD cards |
| astSDList | SD card list |
| udwArrayNum | Number of RAIDs |
| astArrayList | RAID list |
| udwExtendCabinet1HDDNum | Number of HDDs in storage expansion enclosure -1 |
| astExtendCabinet1HDDList | HDD list of storage expansion enclosure-1 |
| udwExtendCabinet2HDDNum | Number of HDDs in storage expansion enclosure -2 |
| astExtendCabinet2HDDList | HDD list of storage expansion enclosure-2 |
| udwNASNum | Number of NAS |
| astNASList | NAS list |
| udweSATANum | Number of eSATA disks |
| asteSATAList | eSATA list |
| udwIPSANNum | Number of IP SANs |
| astIPSANList | IP SAN list |
| byRes | Reserved field |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Storage container info

|  |
| --- |
| typedef struct tagNETDEVStorageContainerInfo  {  UINT32 udwID;  UINT32 udwRemainCapacity;  UINT32 udwTotalCapacity;  CHAR szManufacturer[NETDEV\_LEN\_64];  UINT32 udwStatus;  UINT32 udwProperty;  UINT32 udwFormatProgress;  UINT32 udwGroupID;  UINT32 udwTemperature;  BYTE byRes[124];  }NETDEV\_STORAGE\_CONTAINER\_INFO\_S,\*LPNETDEV\_STORAGE\_CONTAINER\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | HDD ID |
| udwRemainCapacity | Remaining space of storage container (MB) |
| udwTotalCapacity | Total capacity of storage container (MB) |
| szManufacturer | Manufacturer name |
| udwStatus | Storage container status. See [NETDEV\_STORAGE\_CONTAINER\_STATUS\_E](#_存储容器状态). |
| udwProperty | Storage container property. Invalid when udwStatus is 0. See [NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_E](#_存储容器属性). |
| udwFormatProgress | Formatting progress. Percentage |
| udwGroupID | Disk group ID |
| udwTemperature | HDD temperature (°C) |
| byRes | Reserved field |

**See also:**

[NETDEV\_STORAGE\_CONTAINER\_INFO\_LIST\_S](#_存储容器信息列表)

### Expansion storage container info

|  |
| --- |
| typedef struct tagNETDEVExtendStorageContainerInfo  {  UINT32 udwID;  UINT32 udwRemainCapacity;  UINT32 udwTotalCapacity;  UINT32 udwAddressType;  CHAR szIPAddress[NETDEV\_LEN\_64];  CHAR szPath[NETDEV\_LEN\_256];  UINT32 udwUsageType;  UINT32 udwStatus;  UINT32 udwProperty;  UINT32 udwFormatProgress;  UINT32 udwGroupID;  BYTE byRes[128];  }NETDEV\_EXTEND\_STORAGE\_CONTAINER\_INFO\_S,\*LPNETDEV\_EXTEND\_STORAGE\_CONTAINER\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | HDD ID |
| udwRemainCapacity | Remaining space of storage container (MB) |
| udwTotalCapacity | Total capacity of storage container (MB) |
| udwAddressType | IP type. See [NETDEV\_ADDR\_TYPE\_E](#_地址类型). |
| szIPAddress | Server IP |
| szPath | Storage path of NAS server |
| udwUsageType | Usage. See [NETDEV\_STORAGE\_CONTAINER\_USAGE\_TYPE\_E](#_存储容器用途) |
| udwStatus | Storage container status. See [NETDEV\_STORAGE\_CONTAINER\_STATUS\_E](#_存储容器状态). |
| udwProperty | Storage container property. Invalid when udwStatus is 0. See [NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_E](#_存储容器属性). |
| udwFormatProgress | Formatting progress. Percentage |
| udwGroupID | Disk group ID |
| byRes | Reserved field |

**See also:**

[NETDEV\_STORAGE\_CONTAINER\_INFO\_LIST\_S](#_存储容器信息列表)

### HDD S.M.A.R.T. info

|  |
| --- |
| typedef struct tagNETDEVHDDSmartInfo  {  UINT32 udwID;  CHAR szManufacturer[NETDEV\_LEN\_64];  UINT32 udwTemperature;  CHAR szDeviceModel[NETDEV\_LEN\_64];  UINT32 udwUsedDays;  UINT32 udwHealthAssessment;  CHAR szFirmware[NETDEV\_LEN\_64];  UINT32 udwSmartNum;  [NETDEV\_HDD\_SMART\_DETAILS\_INFO\_S](#_硬盘Smart详细信息) SmartDetailsInfoList[NETDEV\_DISK\_SMART\_MAX\_NUM];  BOOL bCheckResult;  UINT32 udwCheckPrograss;  UINT32 udwCheckStatus;  UINT32 udwCheckType;  BYTE byRes[128];  }NETDEV\_HDD\_SMART\_INFO\_S,\*LPNETDEV\_HDD\_SMART\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | IN storage container ID |
| szManufacturer | Manufacturer name |
| udwTemperature | Temperature (°C) |
| szDeviceModel | HDD model |
| udwUsedDays | Number of days used |
| udwHealthAssessment | Health assessment result. See [NETDEV\_HDD\_HEALTH\_ASSESSMENT\_STATUS\_E](#_硬盘健康评估状态). |
| szFirmware | HDD firmware version |
| udwSmartNum | Number of S.M.A.R.T. details. |
| SmartDetailsInfoList | S.M.A.R.T. details list |
| bCheckResult | Check result. TRUE: Pass FALSE: Fail |
| udwCheckPrograss | Check progress: [0,100] |
| udwCheckStatus | Check status. See [NETDEV\_HDD\_SMART\_CHECK\_STATUS\_E](#_硬盘Smart检测状态). |
| udwCheckType | Check type. See [NETDEV\_HDD\_SMART\_CHECK\_TYPE\_E](#_硬盘Smart检测类型). |
| byRes | Reserved field |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### HDD S.M.A.R.T. info

|  |
| --- |
| typedef struct tagNETDEVHDDSmartCheckInfo  {  UINT32 udwID;  UINT32 udwType;  BYTE byRes[128];  }NETDEV\_HDD\_SMART\_CHECK\_INFO\_S,\*LPNETDEV\_HDD\_SMART\_CHECK\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Storage container ID |
| udwType | Check type. See [NETDEV\_HDD\_SMART\_CHECK\_TYPE\_E](#_硬盘Smart检测类型). |
| byRes | Reserved field |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Photo server configuration list

|  |
| --- |
| typedef struct tagNETDEVPhotoServerCfgInfoList  {  UINT32 udwNum;  [NETDEV\_PHOTO\_SERVER\_CFG\_INFO\_S](#_照片接收服务器配置信息)  astPhotoServerList[NETDEV\_PHOTO\_SERVER\_MAX\_NUM];  BYTE byRes[512];  }NETDEV\_PHOTO\_SERVER\_CFG\_INFO\_LIST\_S,\* LPNETDEV\_PHOTO\_SERVER\_CFG\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of photo servers. Currently 2 is supported. |
| astPhotoServerList | Photo server list |
| byRes | Reserved field |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Enhancement configuration info

|  |
| --- |
| typedef struct tagNETDEVEnhancedConfigInfo  {  BOOL bEnabledStorage;  BYTE byRes[512];  }NETDEV\_ENHANCED\_CONFIG\_INFO\_S, \*LPNETDEV\_ENHANCED\_CONFIG\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabledStorage | Enable or disable UNP/cloud device storage. 0: Disable 1: Enable |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Smart server list

|  |
| --- |
| typedef struct tagNETDEVDataServerList  {  UINT32 udwNum;  [NETDEV\_DATA\_SERVER\_INFO](#_智能服务器配置信息) astDataServerList[NETDEV\_INTELLIGENT\_SERVER\_MAX\_NUM];  BYTE byRes[128];  }NETDEV\_DATA\_SERVER\_LIST, \*LPNETDEV\_DATA\_SERVER\_LIST; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of management servers. Currently 2 is supported. |
| astDataServerList | Management server list |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Smart server configuration info

|  |
| --- |
| typedef struct tagNETDEVManagerServerInfo  {  UINT32 udwIndex;  UINT32 udwMngProtocol;  CHAR szDeviceID[NETDEV\_LEN\_64];  CHAR szServerID[NETDEV\_LEN\_64];  CHAR szDeviceName[NETDEV\_DEV\_NAME\_LEN\_MAX];  CHAR szRegPassword[NETDEV\_LEN\_32];  CHAR szVideoChlID[NETDEV\_LEN\_64];  CHAR szAudioChlID[NETDEV\_LEN\_64];  UINT32 udwExpire;  CHAR szAddress[NETDEV\_IPADDR\_STR\_MAX\_LEN];  UINT32 udwPort;  BYTE byRes[256];  }NETDEV\_MANAGER\_SERVER\_INFO\_S, \*LPNETDEV\_MANAGER\_SERVER\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwIndex | Management server index. Starts from 0. |
| udwMngProtocol | Communication protocol. See [NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_E](#_管理服务器接入协议类型). |
| szDeviceID | Frontend device ID. Range: [1, 32] |
| szServerID | Register server ID. Range: [1, 32] |
| szDeviceName | Device name. Range: [0, 20] |
| szRegPassword | Register server password. Range: [0, 16] |
| szVideoChlID | Video channel ID. Range: [0,32] |
| szAudioChlID | Audio channel ID. Range: [0,32] |
| udwExpire | Registration validity period. [3600,36000] seconds. |
| szAddress | Server IP (IPv4) |
| udwPort | Server port |
| byRes | Reserved field. |

**See also:**

[NETDEV\_DATA\_SERVER\_LIST](#_智能服务器配置列表)

### Server user info

|  |
| --- |
| typedef struct tagNETDEVServerUserInfo  {  CHAR szUserName[NETDEV\_USERNAME\_LEN];  BOOL bModifyPasswd;  CHAR szPasswd[NETDEV\_PASSWORD\_LEN];  BYTE byRes[256];  }NETDEV\_SERVER\_USER\_INFO\_S, \*LPNETDEV\_SERVER\_USER\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szUserName | Username. Recommended range: [1,32] |
| bModifyPasswd | Whether it is required to change the password. 0: No 1: Yes |
| szPasswd | Platform access password. Recommended range: [0,32]. An empty string will be passed in if the password is empty. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_MANAGER\_SERVER\_INFO\_S](#_智能服务器配置信息_1)

### Motion detection alarm parameters

|  |
| --- |
| typedef struct tagNETDEVMotionIntervalInfo  {  UINT32 udwSuppressTime;  UINT32 udwClearTime;  BYTE byRes[512];  }NETDEV\_MOTION\_INTERVAL\_INFO\_S, \*LPNETDEV\_MOTION\_INTERVAL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSuppressTime | Alarm suppression time (unit: second). Range: [5, 3600] |
| udwClearTime | Alarm suppression reset time (unit: second). Range: [1,600] |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Management server list

|  |
| --- |
| typedef struct tagNETDEVManagerServerList  {  UINT32 udwNum;  [NETDEV\_MANAGER\_SERVER\_INFO\_S](#_智能服务器配置信息_1)  astManagerServerList[NETDEV\_MANAGER\_SERVER\_MAX\_NUM];  BYTE byRes[128];  }NETDEV\_MANAGER\_SERVER\_LIST\_S, \*LPNETDEV\_MANAGER\_SERVER\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of management servers. Currently 2 is supported. |
| astManagerServerList | Management server list |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Visible range parameters

|  |
| --- |
| typedef struct tagNETDEVVisibleRangeInfo  {  DOUBLE dAzimuthHorizontal;  DOUBLE dAzimuthVertical;  DOUBLE dInclinationHorizontal;  DOUBLE dAngelHorizontal;  DOUBLE dAngelVertical;  DOUBLE dDistance;  DOUBLE dInstallationHeight;  BYTE byRes[248];  }NETDEV\_VISIBLE\_RANGE\_INFO\_S, \*LPNETDEV\_VISIBLE\_RANGE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dAzimuthHorizontal | Horizontal azimuth angle (angle between the central axis of the projection of the visible range on the horizontal plane and the reference axis of the horizontal plane). Unit: radian. Range: [-1.0000,+1.0000] |
| dAzimuthVertical | Vertical azimuth angle (angle between the central axis of the projection of the visible range on the vertical plane and the reference axis of the vertical plane). Unit: radian. Range: [-1.0000,+1.0000] |
| dInclinationHorizontal | Horizontal inclination angle (angle between the horizontal reference plane of the device and the horizontal plane). Unit: radian. Range: [-1.0000,+1.0000] |
| dAngelHorizontal | Horizontal viewing angle (unit: degree, 4 decimal places) |
| dAngelVertical | Vertical viewing angle (unit: degree, 4 decimal places) |
| dDistance | Visual distance (unit: meter, 4 decimal places) |
| dInstallationHeight | Installation height (unit: meter, 4 decimal places) |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Verification info

|  |
| --- |
| typedef struct tagNETDEVResetPwdVertifyInfo  {  CHAR szEMail[NETDEV\_LEN\_256]; /\* Email\*/  CHAR szPhone[NETDEV\_LEN\_256]; /\* Phone number\*/  BYTE byRes[256]; /\* Reserved field\*/  }NETDEV\_RESET\_PWD\_VERTIFY\_INFO\_S, \*LPNETDEV\_RESET\_PWD\_VERTIFY\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szEMail | E-mail |
| szPhone | Phone number |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Storage quota info

|  |
| --- |
| typedef struct tagNETDEVStorageQuotaInfo  {  INT32 dwStorageCapUnit;  INT32 dwRecUsedCap;  INT32 dwPicUsedCap;  INT32 dwAIPicUsedCap;  INT32 dwUsedCap;  INT32 dwTotalCap;  INT32 dwRecQuota;  INT32 dwPicQuota;  INT32 dwAIPicQuota;  INT32 dwGroupID;  INT32 dwGroupTotalCap;  BYTE byRes[256];  }NETDEV\_STORAGE\_QUOTA\_INFO\_S, \*LPNETDEV\_STORAGE\_QUOTA\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwStorageCapUnit | Unit of storage capacity (default is GB). See [NETDEV\_STORAGE\_CAP\_UNIT\_E](#_存储容量单位). |
| dwRecUsedCap | Channel's used capacity for recordings |
| dwPicUsedCap | Channel’s used capacity for snapshots |
| dwAIPicUsedCap | Channel’s used capacity for smart snapshots. Required for IPC. |
| dwUsedCap | Channel’s used storage capacity. Required for IPC. |
| dwTotalCap | Channel’s total storage capacity |
| dwRecQuota | Channel’s capacity quota for recordings |
| dwPicQuota | Channel’s capacity quota for snapshots |
| dwAIPicQuota | Channel’s capacity quota for smart snapshots. Required for IPC. |
| dwGroupID | Disk group ID |
| dwGroupTotalCap | Total capacity of the selected disk group |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Custom protocol info

|  |
| --- |
| typedef struct tagNETDEVCustomProInfo  {  UINT32 udwTransport;  UINT32 udwPort;  [NETDEV\_CUSTON\_PROTOCOL\_SIMPLE\_INFO\_S](#_自定义协议配置信息) stCustomProSimpleInfo;  [NETDEV\_CUSTON\_STREAM\_INFO\_S](#_自定义流协议信息) stMainStream;  [NETDEV\_CUSTON\_STREAM\_INFO\_S](#_自定义流协议信息) stSubStream;  [NETDEV\_CUSTON\_STREAM\_INFO\_S](#_自定义流协议信息) stThirdStream;  BYTE byRes[512];  }NETDEV\_CUSTON\_PROTOCOL\_INFO\_S,\* LPNETDEV\_CUSTON\_PROTOCOL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTransport | Transport mode. See [NETDEV\_TRANS\_PROTOCOL\_E](#_传输协议). |
| udwPort | Port |
| stCustomProSimpleInfo | Custom protocol configuration info |
| stMainStream | Main stream protocol info |
| stSubStream | Sub stream protocol info |
| stThirdStream | Third stream protocol info |
| byRes | Reserved field |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Custom protocol configuration info

|  |
| --- |
| typedef struct tagNETDEVCustomProSimpleInfo  {  UINT32 udwProID;  CHAR szProName[NETDEV\_LEN\_64];  BYTE byRes[128];  }NETDEV\_CUSTON\_PROTOCOL\_SIMPLE\_INFO\_S,\* LPNETDEV\_CUSTON\_PROTOCOL\_SIMPLE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwProID | Custom protocol ID. Required |
| szProName | Custom protocol name. Range: [0, 64] |
| byRes | Reserved field |

**See also:**

[NETDEV\_CUSTON\_PROTOCOL\_INFO\_S](#_自定义协议信息)

### Custom stream protocol info

|  |
| --- |
| typedef struct tagNETDEVCustomStreamInfo  {  UINT32 udwEnable;  CHAR szStreamURL[NETDEV\_LEN\_128];  CHAR szPlaybackStreamURL[NETDEV\_LEN\_128];  BYTE byRes[384];  }NETDEV\_CUSTON\_STREAM\_INFO\_S,\* LPNETDEV\_CUSTON\_STREAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwEnable | Whether is enabled. 1: Yes 0: No |
| szStreamURL | Live stream URL. Range: [0, 128] |
| szPlaybackStreamURL | Playback URL. Range: [0, 128] |
| byRes | Reserved field |

**See also:**

[NETDEV\_CUSTON\_PROTOCOL\_INFO\_S](#_自定义协议信息)

### Simple info of custom protocol

|  |
| --- |
| typedef struct tagNETDEVCustomProSimpleInfoList  {  UINT32 udwNum;  [NETDEV\_CUSTON\_PROTOCOL\_SIMPLE\_INFO\_S](#_自定义协议配置信息_1) astCustomProSimpleList[NETDEV\_LEN\_64];  BYTE byRes[512];  }NETDEV\_CUSTON\_PROTOCOL\_SIMPLE\_INFO\_LIST\_S,\* LPNETDEV\_CUSTON\_PROTOCOL\_SIMPLE\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of custom protocols. Max 64. |
| astCustomProSimpleList | Custom protocol list |
| byRes | Reserved field |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Custom protocol info

|  |
| --- |
| typedef struct tagNETDEVCustomProSimpleInfo  {  UINT32 udwProID;  CHAR szProName[NETDEV\_LEN\_64];  BYTE byRes[128];  }NETDEV\_CUSTON\_PROTOCOL\_SIMPLE\_INFO\_S,\* LPNETDEV\_CUSTON\_PROTOCOL\_SIMPLE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwProID | Custom protocol ID. Required |
| szProName | Custom protocol name. Range: [0, 64] |
| byRes | Reserved field |

**See also:**

[NETDEV\_CUSTON\_PROTOCOL\_SIMPLE\_INFO\_LIST\_S](#_自定义协议配置单例信息列表)

### Storage policy info

|  |
| --- |
| typedef struct tagNETDEVStorageStrategy  {  INT32 dwStorageRes;  [NETDEV\_REC\_STORAGE\_STRATEGY\_INFO\_S](#_视频存储策略信息) stRecStoreStrategyInfo;  [NETDEV\_PIC\_STORAGE\_STRATEGY\_INFO\_S](#_图片存储策略信息) stPicStoreStrategyInfo;  [NETDEV\_PIC\_STORAGE\_STRATEGY\_INFO\_S](#_图片存储策略信息) stAIPicStoreStrategyInfo;  BYTE byRes[512];  }NETDEV\_STORAGE\_STRATEGY\_S, \*LPNETDEV\_STORAGE\_STRATEGY\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwStorageRes | Storage resource. Required for IPC. See [NETDEV\_STORAGE\_RES\_TYPE\_E](#_存储资源). |
| stRecStoreStrategyInfo | Video storage policy info |
| stPicStoreStrategyInfo | Storage policy for normal snapshots |
| stAIPicStoreStrategyInfo | Storage policy for smart snapshots |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Video storage policy info

|  |
| --- |
| typedef struct tagNETDEVRecStorageStrategyInfo  {  BOOL bStoregeEnabled;  INT32 dwFullStrategy;  INT32 dwPlanStreamType;  INT32 dwManualStreamType;  INT32 dwEventStreamType;  BYTE byRes[256];  }NETDEV\_REC\_STORAGE\_STRATEGY\_INFO\_S, \*LPNETDEV\_REC\_STORAGE\_STRATEGY\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bStoregeEnabled | Enable or disable storage. 0: Enable 1: Disable |
| dwFullStrategy | Full strategy policy parameter. Default: Overwrite when storage is full. See [NETDEV\_FULL\_STRATEGY\_TYPE\_E](#_满存储策略). |
| dwPlanStreamType | Stream type of scheduled recording. See [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_实况业务流索引). |
| dwManualStreamType | Stream type of manual recording. See [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_实况业务流索引). |
| dwEventStreamType | Stream type of alarm recording. See [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_实况业务流索引). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_STORAGE\_STRATEGY\_S](#_存储策略信息)

### Image storage policy info

|  |
| --- |
| typedef struct tagNETDEVPicStorageStrategyInfo  {  INT32 dwStorageMode;  INT32 dwFullStrategy;  BYTE byRes[256];  }NETDEV\_PIC\_STORAGE\_STRATEGY\_INFO\_S, \*LPNETDEV\_PIC\_STORAGE\_STRATEGY\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwStorageMode | Storage mode. See [NETDEV\_PIC\_STORAGE\_MODE\_E](#_图片存储模式) |
| dwFullStrategy | Full strategy policy parameter. Default: Overwrite when storage is full. See [NETDEV\_FULL\_STRATEGY\_TYPE\_E](#_满存储策略). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_STORAGE\_STRATEGY\_S](#_存储策略信息)

### Back focus finetune parameters

|  |
| --- |
| typedef struct tagNETDEVBackFocusInfo  {  UINT32 udwCmd;  UINT32 udwPara1;  BYTE byRes[256];  }NETDEV\_BACKFOCUS\_INFO\_S, \*LPNETDEV\_BACKFOCUS\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwCmd | For commands see [NETDEV\_BACKFOCUS\_CMD\_E](#_后焦控制操作命令). |
| udwPara1 | For parameters see [NETDEV\_BACKFOCUS\_CMD\_E](#_后焦控制操作命令). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Smart attribute configuration

|  |
| --- |
| typedef struct tagNETDEVSmartAttrConfig  {  [NETDEV\_FACE\_CFG\_S](#_人脸属性配置) stFaceCfg;  [NETDEV\_TEMP\_CFG\_S](#_温度属性配置) stTempCfg;  BYTE byRes[512];  }NETDEV\_SMART\_ATTR\_CFG\_S, \*LPNETDEV\_SMART\_ATTR\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stFaceCfg | Face attribute configuration |
| stTempCfg | Temperature attribute configuration |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Face attribute configuration

|  |
| --- |
| typedef struct tagNETDEVFaceConfig  {  UINT32 udwCondition;  BYTE byRes[512];  }NETDEV\_FACE\_CFG\_S, LPNETDEV\_FACE\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwCondition | Alarm condition. 0: Not wearing (including Unknown) 1: Wearing |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SMART\_ATTR\_CFG\_S](#_智能属性配置)

### Temperature attribute configuration

|  |
| --- |
| typedef struct tagNETDEVTempConfig  {  UINT32 udwCondition;  CHAR szThreshold[NETDEV\_LEN\_64];  CHAR szdwRange[NETDEV\_LEN\_64];  BYTE byRes[512];  }NETDEV\_TEMP\_CFG\_S, LPNETDEV\_TEMP\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwCondition | Alarm condition. 0: Lower than 1: Higher than 2: Match |
| szThreshold | Threshold temperature (unit: °C, 2 decimal places). |
| szdwRange | Temperature tolerance (unit: °C, 2 decimal places). Range: [0.00, 10.00]. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SMART\_ATTR\_CFG\_S](#_智能属性配置)

### Image correction parameters

|  |
| --- |
| typedef struct tagNETDEV\_ImageCorrectParam  {  BOOL bEnabled;  UINT32 udwFaceWidth;  UINT32 udwPupilDistance;  UINT32 udwMinImageSize;  UINT32 udwMaxImageSize;  FLOAT fBlackAndWhiteThreshold;  BYTE byRes[256];  }NETDEV\_IMAGE\_CORRECT\_PARAM\_S, \*LPNETDEV\_IMAGE\_CORRECT\_PARAM\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Enable or disable. 0: Disable 1: Enable |
| udwFaceWidth | Width of generated face (unit: px). Range: [120,150]. |
| udwPupilDistance | Pupillary distance (unit: px). Range: [120,150]. |
| udwMinImageSize | Minimum image size (unit: KB) |
| udwMaxImageSize | Maximum image size (unit: KB) |
| fBlackAndWhiteThreshold | Black and white threshold. Range: -1 |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Device time synchronization

|  |
| --- |
| typedef struct tagNETDEVTimeSyncMode  {  UINT32 udwMode;  BYTE byRes[256];  }NETDEV\_TIME\_SYNCMODE\_S, \*LPNETDEV\_TIME\_SYNCMODE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwMode | Time synchronization mode. See [NETDEV\_TIME\_SYNCMODE\_E](#_时间同步方式枚举). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Automatic scene switching

|  |
| --- |
| typedef struct tagNETDEVSceneAutoSwitchInfo  {  BOOL bAutoSwitchEnable;  BYTE byRes[256];  }NETDEV\_SCENE\_AUTO\_SWITCH\_INFO\_S, \*LPNETDEV\_SCENE\_AUTO\_SWITCH\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bAutoSwitchEnable | Enable or disable auto switch. 0: Disable 1: Enable |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Scene template info

|  |
| --- |
| typedef struct tagNETDEVSceneTemplateInfo  {  UINT32 udwID;  BYTE byRes[256];  }NETDEV\_SCENE\_TEMPLATE\_INFO\_S, \*LPNETDEV\_SCENE\_TEMPLATE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Scene template ID. Starts from 0. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### All scene info

|  |
| --- |
| typedef struct tagNETDEVSceneInfoList  {  UINT32 udwNum;  [NETDEV\_SCENE\_INFO\_S](#_场景信息) astSceneInfo[NETDEV\_MAX\_SCENE\_INFO\_NUM];  BYTE byRes[256];  }NETDEV\_SCENE\_INFO\_LIST\_S, \*LPNETDEV\_SCENE\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of scenes |
| astSceneInfo | Scene Info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Scene info

|  |
| --- |
| typedef struct tagNETDEVSceneInfo  {  CHAR szName[NETDEV\_LEN\_128];  BOOL bAutoSwitchEnable;  UINT32 udwID;  UINT32 udwType;  UINT32 udwPriority;  UINT32 udwTriggerNum;  [NETDEV\_TRIGGER\_DETAIL\_INFO\_S](#_场景自动切换触发条件详细信息) astTriggerInfo[NETDEV\_MAX\_TRIGGER\_DETAIL\_INFO\_NUM];  BYTE byRes[256];  }NETDEV\_SCENE\_INFO\_S, \*LPNETDEV\_SCENE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szName | Scene name |
| bAutoSwitchEnable | Enable or disable auto switch. 0: Disable 1: Enable |
| udwID | Scene template ID |
| udwType | Scene type |
| udwPriority | Priority level. Range: [1, SceneTemplateNum]. The maximum value is the number of supported scene templates. |
| udwTriggerNum | Number of automatic scene switching triggers |
| astTriggerInfo | Detailed info about automatic scene switching triggers |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SCENE\_INFO\_LIST\_S](#_所有场景信息)

### Detailed info about automatic scene switching triggers

|  |
| --- |
| typedef struct tagNETDEVTriggerDetailInfo  {  CHAR szBegin[NETDEV\_LEN\_16];  CHAR szEnd[NETDEV\_LEN\_16];  UINT32 udwEnvironmentNum;  [NETDEV\_ENV\_PARAM\_INFO\_S](#_环境参数信息) astEnvironmentInfo[NETDEV\_MAX\_ENV\_PARAM\_NUM];  BYTE byRes[256];  }NETDEV\_TRIGGER\_DETAIL\_INFO\_S, \*LPNETDEV\_TRIGGER\_DETAIL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szBegin | Start time |
| szEnd | End time |
| udwEnvironmentNum | Number of environment parameters |
| astEnvironmentInfo | Environment parameter info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SCENE\_INFO\_S](#_场景信息)

### Environment parameter info

|  |
| --- |
| typedef struct tagNETDEVEnvParamInfo  {  UINT32 udwType;  UINT32 udwMin;  UINT32 udwMax;  BYTE byRes[256];  }NETDEV\_ENV\_PARAM\_INFO\_S, \*LPNETDEV\_ENV\_PARAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | Environment parameter type |
| udwMin | Minimum value. Cannot be greater than the minimum value. |
| udwMax | Maximum value. Cannot be less than the minimum value. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_TRIGGER\_DETAIL\_INFO\_S](#_场景自动切换触发条件详细信息)

### Attribute collection info

|  |
| --- |
| typedef struct tagNETDEVAttrCollectInfo  {  BOOL bAttributeAll;  [NETDEV\_FACE\_ATTR\_INFO\_S](#_人脸属性开关信息) stFaceAttrInfo;  [NETDEV\_PED\_ATTR\_INFO\_S](#_行人属性开关信息) stPedAttrInfo;  [NETDEV\_NO\_MOTOR\_VEH\_ATTR\_INFO\_S](#_非机动车属性开关信息_1) stNoMotorVehAttrIofo;  [NETDEV\_MOTOR\_VEH\_ATTR\_INFO\_S](#_车辆属性开关信息_1) stMotorVehAttrInfo;  BYTE byRes[512];  }NETDEV\_ATTR\_COLLECT\_INFO\_S, \*LPNETDEV\_ATTR\_COLLECT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bAttributeAll | Whether are all attributes. 0: No 1: Yes |
| stFaceAttrInfo | Enable or disable face attributes |
| stPedAttrInfo | Enable or disable pedestrian attributes |
| stNoMotorVehAttrIofo | Enable or disable non-motor vehicle attribute |
| stMotorVehAttrInfo | Enable or disable motor vehicle attributes |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Face attributes enable/disable info

|  |
| --- |
| typedef struct tagNETDEVFaceAttrInfo  {  BOOL bIsEnableGender;  BOOL bIsEnableAge;  BOOL bIsEnableGlasses;  BOOL bIsEnableMaskFlag;  BOOL bIsEnableTemp;  BYTE byRes[256];  }NETDEV\_FACE\_ATTR\_INFO\_S, \*LPNETDEV\_FACE\_ATTR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bIsEnableGender | Enable or disable gender detection. 0: Disable 1: Enable |
| bIsEnableAge | Enable or disable age detection. 0: Disable 1: Enable |
| bIsEnableGlasses | Enable or disable glasses detection. 0: Disable 1: Enable |
| bIsEnableMaskFlag | Enable or disable mask detection. 0: Disable 1: Enable |
| bIsEnableTemp | Enable or disable temperature detection. 0: Disable 1: Enable |
| byRes | Reserved field. |

**See also:**

[NETDEV\_ATTR\_COLLECT\_INFO\_S](#_属性采集信息)

### Pedestrian attributes enable/disable info

|  |
| --- |
| typedef struct tagNETDEVPedAttrInfo  {  BOOL bIsEnableGender;  BOOL bIsEnableAge;  BOOL bIsEnableMaskFlag;  BOOL bIsEnableUpperColor;  BOOL bIsEnableLowerColor;  BOOL bIsEnableUpClothesTexture;  BOOL bIsEnableUpperClothingStyle;  BOOL bIsEnableLowerClothingStyle;  BOOL bIsEnableShoes;  BOOL bIsEnableBodyAngle;  BOOL bIsEnableBagInfo;  BOOL bIsEnableHairStyle;  BOOL bIsEnableDirection;  BYTE byRes[256];  }NETDEV\_PED\_ATTR\_INFO\_S, \*LPNETDEV\_PED\_ATTR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bIsEnableGender | Enable or disable gender detection. 0: Disable 1: Enable |
| bIsEnableAge | Enable or disable age detection. 0: Disable 1: Enable |
| bIsEnableMaskFlag | Enable or disable mask detection. 0: Disable 1: Enable |
| bIsEnableUpperColor | Enable or disable coat color detection. 0: Disable 1: Enable |
| bIsEnableLowerColor | Enable or disable trousers color detection. 0: Disable 1: Enable |
| bIsEnableUpClothesTexture | Enable or disable coat texture detection. 0: Disable 1: Enable |
| bIsEnableUpperClothingStyle | Enable or disable upper garment style detection. 0: Disable 1: Enable |
| bIsEnableLowerClothingStyle | Enable or disable lower garment style detection. 0: Disable 1: Enable |
| bIsEnableShoes | Enable or disable shoe detection. 0: Disable 1: Enable |
| bIsEnableBodyAngle | Enable or disable body angle detection. 0: Disable 1: Enable |
| bIsEnableBagInfo | Enable or disable bag detection. 0: Disable 1: Enable |
| bIsEnableHairStyle | Enable or disable hairstyle detection. 0: Disable 1: Enable |
| bIsEnableDirection | Enable or disable movement direction detection. 0: Disable 1: Enable |
| byRes | Reserved field. |

**See also:**

[NETDEV\_ATTR\_COLLECT\_INFO\_S](#_属性采集信息)

### Non-motor vehicle attributes enable/disable info

|  |
| --- |
| typedef struct tagNETDEVNoMotorVehAttrInfo  {  BOOL bIsEnableGender;  BOOL bIsEnableAge;  BOOL bIsEnableUpperColor;  BOOL bIsEnableUpperClothingStyle;  BOOL bIsEnableNonVehicleType;  BOOL bIsEnableDirection;  BOOL bIsEnableSpeed;  BOOL bEnablePlateNumber;  BOOL bEnablePlateType;  BOOL bEnablePlateColor;  BOOL bEnableLaneNo;  BOOL bEnablePeccancy;  BYTE byRes[256];  }NETDEV\_NO\_MOTOR\_VEH\_ATTR\_INFO\_S, \*LPNETDEV\_NO\_MOTOR\_VEH\_ATTR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bIsEnableGender | Enable or disable gender detection. 0: Disable 1: Enable |
| bIsEnableAge | Enable or disable age detection. 0: Disable 1: Enable |
| bIsEnableUpperColor | Enable or disable coat color detection. 0: Disable 1: Enable |
| bIsEnableUpperClothingStyle | Enable or disable upper garment style detection. 0: Disable 1: Enable |
| bIsEnableNonVehicleType | Enable or disable non-motor vehicle type detection. 0: Disable 1: Enable |
| bIsEnableDirection | Enable or disable driving direction detection. 0: Disable 1: Enable |
| bIsEnableSpeed | Enable or disable speed detection. 0: Disable 1: Enable |
| bEnablePlateNumber | Enable or disable plate number detection. 0: Disable 1: Enable |
| bEnablePlateType | Enable or disable plate type detection. 0: Disable 1: Enable |
| bEnablePlateColor | Enable or disable plate color detection. 0: Disable 1: Enable |
| bEnableLaneNo | Enable or disable lane ID detection. 0: Disable 1: Enable |
| bEnablePeccancy | Enable or disable violation type detection. 0: Disable 1: Enable |
| byRes | Reserved field. |

**See also:**

[NETDEV\_ATTR\_COLLECT\_INFO\_S](#_属性采集信息)

### Motor vehicle attributes enable/disable info

|  |
| --- |
| typedef struct tagNETDEVMotorVehAttrInfo  {  BOOL bIsEnableLogo;  BOOL bIsEnableVehicleType;  BOOL bIsEnableVehicleColor;  BOOL bIsEnablePlateNumber;  BOOL bIsEnablePlateType;  BOOL bIsEnablePlateColor;  BOOL bIsEnableDirection;  BOOL bIsEnableSpeed;  BOOL bEnableSunVisor;  BOOL bEnableSeatBelt;  BOOL bEnableDriverMobile;  BOOL bEnableAim;  BOOL bEnablePendant;  BOOL bEnableLaneNo;  BOOL bEnablePeccancy;  BOOL bEnableYellowPlateMark;  BOOL bEnableVehicleBrandType;  BOOL bEnableDangerousGoodsMark;  BYTE byRes[256];  }NETDEV\_MOTOR\_VEH\_ATTR\_INFO\_S, \*LPNETDEV\_MOTOR\_VEH\_ATTR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bIsEnableLogo | Enable or disable vehicle brand detection. 0: Disable 1: Enable |
| bIsEnableVehicleType | Enable or disable vehicle type detection. 0: Disable 1: Enable |
| bIsEnableVehicleColor | Enable or disable vehicle color detection. 0: Disable 1: Enable |
| bIsEnablePlateNumber | Enable or disable plate number detection. 0: Disable 1: Enable |
| bIsEnablePlateType | Enable or disable plate type detection. 0: Disable 1: Enable |
| bIsEnablePlateColor | Enable or disable plate color detection. 0: Disable 1: Enable |
| bIsEnableDirection | Enable or disable driving detection detection. 0: Disable 1: Enable |
| bIsEnableSpeed | Enable or disable driving speed detection. 0: Disable 1: Enable |
| bEnableSunVisor | Enable or disable sun visor detection. 0: Disable 1: Enable |
| bEnableSeatBelt | Enable or disable seat belt detection. 0: Disable 1: Enable |
| bEnableDriverMobile | Enable or disable driver’s phone usage detection. 0: Disable 1: Enable |
| bEnableAim | Enable or disable inspection sticker detection. 0: Disable 1: Enable |
| bEnablePendant | Enable or disable pendant detection. 0: Disable 1: Enable |
| bEnableLaneNo | Enable or disable lane ID detection. 0: Disable 1: Enable |
| bEnablePeccancy | Enable or disable violation type detection. 0: Disable 1: Enable |
| bEnableYellowPlateMark | Enable or disable yellow plate mark detection. 0: Disable 1: Enable |
| bEnableVehicleBrandType | Enable or disable brand type detection. 0: Disable 1: Enable |
| bEnableDangerousGoodsMark | Enable or disable dangerous goods mark detection. 0: Disable 1: Enable |
| byRes | Reserved field. |

**See also:**

[NETDEV\_ATTR\_COLLECT\_INFO\_S](#_属性采集信息)

### Time configuration

|  |
| --- |
| typedef struct tagNETDEVSystemTimeInfo  {  [NETDEV\_TIME\_ZONE\_E](#_时区) dwTimeZone;  [NETDEV\_TIME\_S](#_时间参数) stTime;  BOOL bEnableDST;  [NETDEV\_TIME\_DST\_CFG\_S](#_时间配置) stTimeDSTCfg;  UINT32 udwDateFormat;  UINT32 udwHourFormat;  BYTE byRes[212];  }NETDEV\_TIME\_CFG\_S, \*LPNETDEV\_TIME\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwTimeZone | Time zone. See [NETDEV\_TIME\_ZONE\_E](#_时区). |
| stTime | Time |
| bEnableDST | Enable or disable DST |
| stTimeDSTCfg | DST configuration |
| udwDateFormat | Date format. 0: YYYY-MM-DD 1: MM-DD-YYYY 2: DD-MM-YYYY |
| udwHourFormat | Time format. 0: 12H format 1: 24H format |
| byRes | Reserved field. |

### Time parameters

|  |
| --- |
| typedef struct tagNETDEVTime  {  INT32 dwYear;  INT32 dwMonth;  INT32 dwDay;  INT32 dwHour;  INT32 dwMinute;  INT32 dwSecond;  }NETDEV\_TIME\_S, \*LPNETDEV\_TIME\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwYear | Year |
| dwMonth | Month |
| dwDay | Day |
| dwHour | Hour |
| dwMinute | Minute |
| dwSecond | Second |

**See also:**

[NETDEV\_TIME\_CFG\_S](#_时间配置_3)

### System IP address

|  |
| --- |
| typedef struct tagSysemIPAddrInfo  {  BOOL bEnabled;  INT64 ulAddressType;  CHAR szIPAddress[NETDEV\_LEN\_64];  CHAR szDomainName[NETDEV\_LEN\_64];  INT64 ulPort;  INT64 ulSynchronizeInterval;  BYTE byRes[128];  }NETDEV\_SYSTEM\_IPADDR\_INFO\_S, \*LPNETDEV\_SYSTEM\_IPADDR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Enable or disable NTP server. 0: Disable 1: Enable |
| ulAddressType | Address type. 0: IPv4 1: IPv6 (currently not in use) 2: Domain name (applicable to NVR and VMS) |
| szIPAddress | NTP server IP. Length: [0, 64]. Required when ulAddressType is 0. |
| szDomainName | NTP server domain name. Length: [0, 0,64]. Required when ulAddressType is 2. |
| ulPort | NTP port. Range: [1-65535]. Not applicable to IPC. |
| ulSynchronizeInterval | Update Interval. Valid range of NVR and VMS: 5/10/15/30m, 1/2/3/6/12h, 1 day, 1 week Valid range of IPC: 30-3600s. The time intervals need to be converted to time values in unit of seconds. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SYSTEM\_NTP\_INFO\_LIST\_S](#_NTP列表)

### Position relative to video wall

|  |
| --- |
| typedef struct tagNETDEVXWVirtualLEDArea  {  [NETDEV\_XW\_COORD\_S](#_坐标信息) stTopLeft;  [NETDEV\_XW\_COORD\_S](#_坐标信息) stBottomRight;  }NETDEV\_XW\_AREA\_S, \*LPNETDEV\_XW\_AREA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stTopLeft | Coordinates of top left corner |
| stBottomRight | Coordinates of bottom right corner |

**See also:**

[NETDEV\_TEMP\_DETECTION\_BLACKBODY\_INFO\_S](#_温度检测黑体信息结构体)

### Coordinates info

|  |
| --- |
| typedef struct tagNETDEVXWArea  {  UINT32 udwX;  UINT32 udwY;  }NETDEV\_XW\_COORD\_S, \*LPNETDEV\_XW\_COORD\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwX | x-coordinate |
| udwY | y-coordinate |

**See also:**

[NETDEV\_XW\_AREA\_S](#_相对于电视墙的位置信息)

### Detailed HDD S.M.A.R.T. info

|  |
| --- |
| typedef struct tagNETDEVHDDSmartDetailsInfo  {  UINT32 udwAttributeID;  CHAR szAttributeName[NETDEV\_LEN\_64];  UINT32 udwStatus;  UINT32 udwHex;  UINT32 udwThresh;  UINT32 udwCurrentValue;  UINT32 udwWorstValue;  UINT32 udwActualValue;  BYTE byRes[128];  }NETDEV\_HDD\_SMART\_DETAILS\_INFO\_S,\*LPNETDEV\_HDD\_SMART\_DETAILS\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwAttributeID | Attribute ID |
| szAttributeName | Attribute name |
| udwStatus | Status. See [NETDEV\_HDD\_SMART\_ASSESSMENT\_STATUS\_E](#_硬盘Smart评估状态). |
| udwHex | Displayed in hexadecimal |
| udwThresh | Threshold |
| udwCurrentValue | Current value |
| udwWorstValue | Worst value |
| udwActualValue | Actual value |
| byRes | Reserved field |

**See also:**

[NETDEV\_HDD\_SMART\_INFO\_S](#_硬盘Smart信息)

### Photo server configuration info

|  |
| --- |
| typedef struct tagNETDEVPhotoServerCfgInfo  {  UINT32 udwProtocol;  UINT32 udwSubProtocol;  UINT32 udwPort;  CHAR szIPV4Address[NETDEV\_IPADDR\_STR\_MAX\_LEN];  CHAR szTollgateID[NETDEV\_LEN\_64];  CHAR szCameraID[NETDEV\_LEN\_64];  BYTE byRes[128];  }NETDEV\_PHOTO\_SERVER\_CFG\_INFO\_S,\* LPNETDEV\_PHOTO\_SERVER\_CFG\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwProtocol | Communication protocol. See [NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_E](#_照片服务器通信协议类型). |
| udwSubProtocol | Communication sub protocol. See [NETDEV\_PHOTO\_SERVER\_SUBPRO\_TYPE\_E](#_照片服务器通信子协议类型). |
| udwPort | TMS server port number: [0,65535] |
| szIPV4Address | TMS server IP (IPv4) |
| szTollgateID | Checkpoint ID (0, 36]. Optional. When absent, it means the field will not be updated. |
| szCameraID | Camera ID (0, 36]. Optional. When absent, it means the field will not be updated. |
| byRes | Reserved field |

**See also:**

[NETDEV\_PHOTO\_SERVER\_CFG\_INFO\_LIST\_S](#_照片接收服务器配置信息列表)

### Smart server configuration info

|  |
| --- |
| typedef struct tagNETDEVDataServerInfo  {  UINT32 udwIndex;  BOOL bEnable;  UINT32 udwProtocol;  CHAR szCameraID[NETDEV\_LEN\_128];  CHAR szTollgateID[NETDEV\_LEN\_128];  CHAR szServerAddr[NETDEV\_LEN\_32];  UINT32 udwServerPort;  CHAR szDrivewayID[NETDEV\_LEN\_64];  CHAR szDirectionID[NETDEV\_LEN\_64];  CHAR szDeviceID[NETDEV\_LEN\_64];  CHAR szAreaId[NETDEV\_LEN\_64];  [NETDEV\_SERVER\_USER\_INFO\_S](#_服务器用户信息) stServerUserInfo;  NETDEV\_VIID\_CFG\_S stViidCfg;  BYTE byRes[148];  }NETDEV\_DATA\_SERVER\_INFO, \*LPNETDEV\_DATA\_SERVER\_INFO; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwIndex | Smart server index. Starts from 0. |
| bEnable | Enable or disable smart server |
| udwProtocol | Communication protocol. Currently only 1: TCP is supported. See [NETDEV\_DATA\_SERVER\_PROTOCOL\_TYPE\_E](#_智能服务器接入协议类型). |
| szCameraID | Camera ID (0, 36]. It means no modification when the length is 0. |
| szTollgateID | Checkpoint ID (0, 36]. It means no modification when the length is 0. |
| szServerAddr | Server IP (IPv4) |
| udwServerPort | Server port |
| szDrivewayID | Road ID. Applicable to IPC. |
| szDirectionID | Direction ID. Applicable to IPC. |
| szDeviceID | Device ID. Applicable to IPC. |
| szAreaId | Area ID. Applicable to IPC. |
| stServerUserInfo | User info |
| stViidCfg | VIID info. See [NETDEV\_VIID\_CFG\_S](#_视图库信息结构体). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_DATA\_SERVER\_LIST](#_智能服务器配置列表)

### All OSD contents.

|  |
| --- |
| typedef struct tagNETDEVOsdContent  {  UINT32 udwNum;  [NETDEV\_OSD\_CONTENT\_INFO\_S](#_通道OSD内容信息) astContentList[NETDEV\_LEN\_32];  }NETDEV\_OSD\_CONTENT\_S,\*LPNETDEV\_OSD\_CONTENT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of OSD areas |
| astContentList | OSD area content list |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### OSD content info

|  |
| --- |
| typedef struct tagNETDEVOSDContentInfo  {  BOOL bEnabled;  UINT32 udwOSDID;  UINT32 udwAreaOSDNum;  UINT32 udwTopLeftX;  UINT32 udwTopLeftY;  [NETDEV\_CONTENT\_INFO\_S](#_内容信息) astContentInfo[NETDEV\_LEN\_8];  }NETDEV\_OSD\_CONTENT\_INFO\_S,\*LPNETDEV\_OSD\_CONTENT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Enable OSD area |
| udwOSDID | OSD area ID. Range: [0,7] |
| udwAreaOSDNum | Number of OSDs in current area |
| udwTopLeftX | x-coordinate of OSD area. Range: [0,9999] |
| udwTopLeftY | y-coordinate of OSD area. Range: [0,9999] |
| astContentInfo | OSD content in current area |

**See also:**

[NETDEV\_OSD\_CONTENT\_S](#_通道OSD所有内容)

### Content info

|  |
| --- |
| typedef struct tagNETDEVContentInfo  {  UINT32 udwContentType;  CHAR szOSDText[NETDEV\_OSD\_TEXT\_MAX\_LEN];  }NETDEV\_CONTENT\_INFO\_S,\*LPNETDEV\_CONTENT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwContentType | OSD content type. See [NETDEV\_OSD\_CONTENT\_TYPE\_E](#_OSD内容类型). |
| szOSDText | OSD text info |

**See also:**

[NETDEV\_OSD\_CONTENT\_INFO\_S](#_通道OSD内容信息)

### Temperature detection alarm configuration info

|  |
| --- |
| typedef struct tagNETDEVTempDetectionRuleInfo  {  UINT32 udwType;  [NETDEV\_TEMP\_DETECTION\_BASE\_RULE\_INFO\_S](#_温度检测基本信息结构体) stEntireImageRuleInfo;  [NETDEV\_TEMP\_DETECTION\_COMM\_RULE\_LIST\_S](#_温度检测通用信息列表) stCommonRuleInfo;  [NETDEV\_TEMP\_DETECTION\_COMP\_RULE\_LIST\_S](#_温度检测比较信息列表) stCompareRuleInfo;  BYTE byRes[512];  }NETDEV\_TEMP\_DETECTION\_RULE\_INFO\_S, \*LPNETDEV\_TEMP\_DETECTION\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | Rule type: 0 Full screen 1: Common 2: Common, comparison 3: Face temperature detection |
| stEntireImageRuleInfo | Full-screen rule info. Required when Type is 0. |
| stCommonRuleInfo | Common rule info. Required when Type is 1 or 2. |
| stCompareRuleInfo | Comparison rule info. Required when Type is 2. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### Structure of basic info about temperature detection

|  |
| --- |
| typedef struct tagNETDEVTemperatureDetectionBasicRuleInfo  {  UINT32 udwType;  INT32 dwCondition;  FLOAT fThreshold;  FLOAT fChangeRate;  FLOAT fRange;  UINT32 udwDuration;  BYTE byRes[512];  }NETDEV\_TEMP\_DETECTION\_BASE\_RULE\_INFO\_S,\*LPNETDEV\_TEMP\_DETECTION\_BASE\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | Alarm type. 0: Highest temperature 1: Lowest temperature 2: Average temperature 3: Temperature difference 4: Temperature change rate (Note: 1. For comparison rules, the alarm type only supports 0, 1, and 2. 2. When the area type is “point”, the alarm type only supports 2 and 4). |
| dwCondition | Alarm condition. 0: Lower than 1: Higher than 2: Match |
| fThreshold | Temperature threshold (unit: degree, 2 decimal places). Range: [-20,150]. Default: 0. Required when alarm type is not “temperature change rate”). |
| fChangeRate | Temperature change rate. Range: [-170,170]. Unit: degree/minute (required when alarm type is “temperature change rate”). Precision: 2 decimal places. |
| fRange | Temperature error tolerance (unit: degree). Range: [0,10]. Precision: 2 decimal places. |
| udwDuration | Duration (unit: second). Range: [1,100] |
| byRes | Reserved field. |

**See also:**

[NETDEV\_TEMP\_DETECTION\_RULE\_INFO\_S](#_温度检测告警的配置信息)

### Temperature detection common rule list

|  |
| --- |
| typedef struct tagNETDEVTempDetectionCommonRuleList  {  UINT32 udwNum;  [NETDEV\_TEMP\_DETECTION\_COMM\_RULE\_INFO\_S](#_温度检测通用信息结构体_1) astTempCommRuleInfo[NETDEV\_LEN\_16];  BYTE byRes[512];  }NETDEV\_TEMP\_DETECTION\_COMM\_RULE\_LIST\_S,\*LPNETDEV\_TEMP\_DETECTION\_COMM\_RULE\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of temperature detection common rules. Starts from 0. Range: [0,12] |
| astTempCommRuleInfo | Structure of general info about temperature detection |
| byRes | Reserved field. |

**See also:**

[NETDEV\_TEMP\_DETECTION\_RULE\_INFO\_S](#_温度检测告警的配置信息)

### Structure of general info about temperature detection

|  |
| --- |
| typedef struct tagNETDEVTempDetectionCommonRuleInfo  {  UINT32 udwID;  BOOL bEnabled;  CHAR szName[NETDEV\_LEN\_512];  UINT32 udwAreaType;  UINT32 udwAreaPointNum;  [NETDEV\_POINT\_S](#_点坐标结构体) astAreaPoint[NETDEV\_LEN\_16];  [NETDEV\_TEMP\_DETECTION\_BASE\_RULE\_INFO\_S](#_温度检测基本信息结构体) stTempBaseInfo;  BYTE byRes[512];  }NETDEV\_TEMP\_DETECTION\_COMM\_RULE\_INFO\_S,\*LPNETDEV\_TEMP\_DETECTION\_COMM\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Rule ID. Starts from 0. Range: [0,5]. Required when using the GET or PUT method. |
| bEnabled | Enable or disable. 0: Disable 1: Enable |
| szName | Rule name. Range: [0,20] |
| udwAreaType | Detection area type. 0: Line 4: Polygon (max hexagon, consistent with smart detection box) 7: Dot |
| udwAreaPointNum | Number of vertices of the detection area 1: Dot 2: Straight line 3-6: Polygon |
| astAreaPoint | Coordinates of the vertices of the detection area |
| stTempBaseInfo | Structure of basic info about temperature detection |
| byRes | Reserved field. |

**See also:**

[NETDEV\_TEMP\_DETECTION\_COMM\_RULE\_LIST\_S](#_温度检测通用信息列表)

### Temperature detection comparison rule list

|  |
| --- |
| typedef struct tagNETDEVTempDetectionCompareRuleList  {  UINT32 udwNum;  [NETDEV\_TEMP\_DETECTION\_COMP\_RULE\_INFO\_S](#_温度检测比较信息结构体) astTempCompareRuleInfo[NETDEV\_LEN\_16];  BYTE byRes[512];  }NETDEV\_TEMP\_DETECTION\_COMP\_RULE\_LIST\_S,\*LPNETDEV\_TEMP\_DETECTION\_COMP\_RULE\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of temperature detection comparison rules. Starts from 0. Range: [0,12] |
| astTempCompareRuleInfo | Structure of comparison info about temperature detection |
| byRes | Reserved field. |

**See also:**

[NETDEV\_TEMP\_DETECTION\_RULE\_INFO\_S](#_温度检测告警的配置信息)

### Structure of comparison info about temperature detection

|  |
| --- |
| typedef struct tagNETDEVTempDetectionCompareRuleInfo  {  UINT32 udwID;  BOOL bEnabled;  CHAR szName[NETDEV\_LEN\_512];  UINT32 udwFirstCompareRuleID;  UINT32 udwSecondCompareRuleID;  [NETDEV\_TEMP\_DETECTION\_BASE\_RULE\_INFO\_S](#_温度检测基本信息结构体) stTempBaseInfo;  BYTE byRes[512];  }NETDEV\_TEMP\_DETECTION\_COMP\_RULE\_INFO\_S,\*LPNETDEV\_TEMP\_DETECTION\_COMP\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Rule ID. Starts from 0. Range: [0,5]. Required when using the GET or PUT method. |
| bEnabled | Enable or disable. 0: Disable 1: Enable |
| szName | Rule name. Range: [0,20] |
| udwFirstCompareRuleID | ID of the first comparison rule. Range: [1,12] |
| udwSecondCompareRuleID | ID of the second comparison rule. Range: [1,12]. The ID must be different from that of the first rule. |
| stTempBaseInfo | Structure of basic info about temperature detection |
| byRes | Reserved field. |

**See also:**

[NETDEV\_TEMP\_DETECTION\_COMP\_RULE\_LIST\_S](#_温度检测比较信息列表)

### Basic device info

|  |
| --- |
| typedef struct tagstNETDEVDeviceBasicInfo  {  [NETDEV\_IPADDR\_INFO\_S](#_IP地址信息) stDevAddr;  [NETDEV\_USER\_SIMPLE\_INFO\_S](#_用户账号信息) stDevUserInfo;  CHAR szDevName[NETDEV\_NAME\_MAX\_LEN];  CHAR szDevDesc[NETDEV\_DESCRIBE\_MAX\_LEN];  CHAR szDevModel[NETDEV\_CODE\_STR\_MAX\_LEN];  INT32 dwDevID;  INT32 dwDevStatus;  INT32 dwDevType;  INT32 dwDevSubType;  INT32 dwOrgID;  INT32 dwAccessProtocol;  INT32 dwAccessMode;  INT32 dwServerID;  INT32 dwAudioResID;  INT32 dwIsPTZNeeded ;  CHAR szVIIDCode[NETDEV\_VIID\_CODE\_LEN];  INT32 dwVIIDStatus;  CHAR szSerialNum[NETDEV\_LEN\_64];  CHAR szSoftVersion[NETDEV\_LEN\_128];  CHAR szMacAddr[NETDEV\_LEN\_32];  INT32 dwStoreStatus;  [NETDEV\_ONVIF\_INFO\_S](#_Onvif信息_1) stOnvifInfo;  [NETDEV\_GBINFO\_S](#_国标信息_1) stGBInfo;  [LPNETDEV\_SMART\_LOCK\_INFO\_S](#_智能锁信息_1) pstSmartLockInfo;  CHAR szManufacture[NETDEV\_LEN\_64];  CHAR szDeviceCode[NETDEV\_LEN\_32];  [LPNETDEV\_IPM\_PLAYER\_BASIC\_INFO\_S](#_播放盒信息_1) pstPlayerInfo;  UINT32 udwCustomProtocolID;  UINT32 udwChlMaxNum;  UINT32 udwChlIndexNum;  UINT32\* pudwChlIndexList;  INT32 dwImageProtocol;  }NETDEV\_DEV\_BASIC\_INFO\_S, \*LPNETDEV\_DEV\_BASIC\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stDevAddr | Device IP address info |
| stDevUserInfo | Device username and password |
| szDevName | Device name |
| szDevDesc | Device description |
| szDevModel | Device model |
| dwDevID | Device IP |
| dwDevStatus | Device status. See [NETDEV\_DEVICE\_STATUS\_E](#_设备状态). |
| dwDevType | Device type. See [NETDEV\_DEVICE\_MAIN\_TYPE\_E](#_设备类型枚举定义). |
| dwDevSubType | Device sub type. See [NETDEV\_DEVICE\_SUB\_TYPE\_E](#_设备子类型). |
| dwOrgID | Organization ID |
| dwAccessProtocol | Connection protocol. See [NETDEV\_ACCESS\_PROTOCOL\_E](#_接入协议). |
| dwAccessMode | Connection mode. See [NETDEV\_DEVICE\_ACCESS\_MODE\_E](#_接入方式). |
| dwServerID | Server ID |
| dwAudioResID | Audio channel ID |
| dwIsPTZNeeded | Whether PTZ is needed. 0: No 1: Yes 255: Adaptive |
| szVIIDCode | VIID ID. Valid for checkpoint device only. |
| dwVIIDStatus | VIID status. Indicates whether device is connected by VIID protocol. 0: Offline 1: Online |
| szSerialNum | Device serial number |
| szSoftVersion | Firmware version |
| szMacAddr | MAC address |
| dwStoreStatus | Device storage status 0: Idle 1: Not formatted 2: Format in progress 3: Running |
| stOnvifInfo | Onvif info |
| stGBInfo | GB info. Required only when AccessProtocol is 3. |
| pstSmartLockInfo | Lock device info. Memory needs to be allocated by the caller. |
| szManufacture | Manufacturer name |
| szDeviceCode | Device ID. [1,32]. Required when adding a player. |
| pstPlayerInfo | Player info. Required when Type is 11. Memory needs to be allocated by malloc. |
| udwCustomProtocolID | Custom protocol ID. Required when AccessProtocol is 4. |
| udwChlMaxNum | Maximum number of device channel IDs. Required when AccessProtocol is 4. |
| udwChlIndexNum | Number of device channel IDs. Required when AccessProtocol is 4. Max 255. |
| pudwChlIndexList | Channel ID list. RAM needs to be allocated dynamically. 256 is recommended. |
| dwImageProtocol | Image protocol. Required when device type (Type) is 5 (smart device). 1: Private 2: VIID |

**See also:**

[NETDEV\_FindDevList](#_通过设备类型查询设备列表), [NETDEV\_FindNextDevInfo](#_逐个获取查找到的设备信息), [NETDEV\_FindCloseDevInfo](#_关闭查找设备信息，释放资源)

### IP address info

|  |
| --- |
| typedef struct tagstNETDEVIPAddrInfo  {  INT32 dwType;  CHAR szIPAddr[NETDEV\_MAX\_URL\_LEN];  INT32 dwPort;  BYTE byRes[60];  }NETDEV\_IPADDR\_INFO\_S, \*LPNETDEV\_IPADDR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwType | Address type. See [NETDEV\_IP\_ADDRESS\_TYPE\_E](#_IP地址类型). |
| szIPAddr | IP address/domain name |
| dwPort | Port number |
| byRes | Reserved field. |

**See also:**

[NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1)

### User account info

|  |
| --- |
| typedef struct tagstNETDEVUserSimpleInfo  {  CHAR szUserName[NETDEV\_USERNAME\_LEN];  CHAR szPassword[NETDEV\_PASSWORD\_LEN];  BYTE byRes[64];  }NETDEV\_USER\_SIMPLE\_INFO\_S, \*LPNETDEV\_USER\_SIMPLE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szUserName | Username |
| szPassword | Password |
| byRes | Reserved field. |

**See also:**

[NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1)

### Onvif info

|  |
| --- |
| typedef struct tagstNETDEVOnvifInfo  {  UINT32 udwTransportMode;  BYTE byRes[128];  }NETDEV\_ONVIF\_INFO\_S, \*LPNETDEV\_ONVIF\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTransportMode | Transport mode. See [NETDEV\_TRANS\_PROTOCOL\_E](#_传输协议). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1)

### GB info

|  |
| --- |
| typedef struct tagNETDEVGBInfo  {  CHAR szUniCode[NETDEV\_LEN\_32];  UINT32 udwTransport;  BYTE byRes[128];  }NETDEV\_GBINFO\_S, \*LPNETDEV\_GBINFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szUniCode | GB resource ID. Range: [1, 32] |
| udwTransport | Transport Mode. 0: TCP 1: UDP |
| byRes | Reserved field |

**See also:**

[NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1)

### Smart lock info

|  |
| --- |
| typedef struct tagNETDEVSmartLockInfo  {  UINT32 udwType;  UINT32 udwSignal;  UINT32 udwStatus;  UINT32 udwBatteryPercent;  CHAR szSN[NETDEV\_LEN\_64];  CHAR szIMEI[NETDEV\_LEN\_64];  CHAR szVersion[NETDEV\_LEN\_256];  CHAR szRoomName[NETDEV\_LEN\_480];  BYTE byRes[128];  }NETDEV\_SMART\_LOCK\_INFO\_S,\*LPNETDEV\_SMART\_LOCK\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | Lock type. 0: Wi-Fi lock 1: NBIoT lock |
| udwSignal | Lock signal. See [NETDEV\_LOCK\_SIGNAL\_E](#_锁信号). |
| udwStatus | Lock status. 0: Online 1: Offline |
| udwBatteryPercent | Battery usage. Range: [0,100] |
| szSN | Lock device serial number. Range: [0,20]. |
| szIMEI | IMEI. Required when Type is 1. Length: [1,16] |
| szVersion | Lock version info. Length: [1,64] |
| szRoomName | Room name. Length: [1, 128] |
| byRes | Reserved field |

**See also:**

[NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1)

### Player info

|  |
| --- |
| typedef struct tagNETDEVIPMPlayerBasicInfo  {  UINT32 udwPlayerID;  UINT32 udwVendor;  UINT32 udwModel;  CHAR szPlayerName[NETDEV\_LEN\_128];  UINT32 udwOrgID;  CHAR szOrgName[NETDEV\_LEN\_128];  CHAR szPlayerMngtServerName[NETDEV\_LEN\_128];  CHAR szPlayerMngtServerIP[NETDEV\_LEN\_32];  [NETDEV\_XW\_RESOLUTION\_S](#_分辨率信息) stResolution;  BYTE byRes[256];  }NETDEV\_IPM\_PLAYER\_BASIC\_INFO\_S,\*LPNETDEV\_IPM\_PLAYER\_BASIC\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPlayerID | Player ID |
| udwVendor | Required when adding players by vendor |
| udwModel | Required when adding players by model |
| szPlayerName | Player name. Range: [1,64] |
| udwOrgID | Organization ID. Required when getting player info. |
| szOrgName | Organization name. Range: [1,64]. Required when getting player info. |
| szPlayerMngtServerName | Player management device name. Range: [1,64]. Required when getting player info. |
| szPlayerMngtServerIP | Player management device IP. Required when getting player info. |
| stResolution | Screen resolution |
| byRes | Reserved field. |

**See also:**

[NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1)

### Resolution information

|  |
| --- |
| typedef struct tagNETDEVXWVideoResolution  {  UINT32 udwWidth;  UINT32 udwHeight;  }NETDEV\_XW\_RESOLUTION\_S, \*LPNETDEV\_XW\_RESOLUTION\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwWidth | Width |
| udwHeight | Height |

**See also:**

[NETDEV\_IPM\_PLAYER\_BASIC\_INFO\_S](#_播放盒信息_1)

### Device channel info

|  |
| --- |
| typedef struct tagNETDEVChlDeviceInfo  {  INT32 dwChannelID;  CHAR szDevModel[NETDEV\_LEN\_64];  CHAR szSerialNum[NETDEV\_LEN\_64];  CHAR szFirmwareVersion[NETDEV\_LEN\_64];  CHAR szHardewareID[NETDEV\_LEN\_64];  CHAR szUbootVersion[NETDEV\_LEN\_64];  BYTE byRes[512];  }NETDEV\_DEVICE\_CHL\_INFO\_S, \*LPNETDEV\_DEVICE\_CHL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| szDevModel | Device model |
| szSerialNum | Hardware serial number |
| szFirmwareVersion | Firmware version |
| szHardewareID | Hardware ID |
| szUbootVersion | UBOOT version |
| byRes | Reserved field |

**See also:**

[NETDEV\_FindDevChlBasicInfoList](#_获取通道设备基本信息列表), [NETDEV\_FindNextChlDeviceInfo](#_逐个获取通道设备基本信息),

[NETDEV\_FindCloseDevChlBasicInfo](#_关闭获取通道设备基本信息)

### Channel info about Xware device’s local signal source

|  |
| --- |
| typedef struct tagstNETDEVDevChnXWEncodeInfo  {  [NETDEV\_DEV\_CHN\_BASE\_INFO\_S](#_设备通道基本信息_1) stChnBaseInfo;  INT32 dwVideoSrc;  BYTE byRes[260];  }NETDEV\_DEV\_CHN\_XW\_ENCODE\_INFO\_S, \*LPNETDEV\_DEV\_CHN\_XW\_ENCODE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stChnBaseInfo | Basic channel info |
| dwVideoSrc | Video input format. See [NETDEV\_VIDEO\_SRC\_TYPE\_E](#_视频输入类型枚举). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CHN\_TYPE\_E](#_通道类型)

### Basic channel info of device

|  |
| --- |
| typedef struct tagstNETDEVDevChnBaseInfo  {  INT32 dwChannelID;  INT32 dwDevID;  INT32 dwOrgID;  INT32 dwChnType;  INT32 dwChnStatus;  INT32 dwChnIndex;  CHAR szChnName[NETDEV\_NAME\_MAX\_LEN];  UINT32 udwRight;  UINT32 udwAbnormalReason;  BYTE byRes[124];  }NETDEV\_DEV\_CHN\_BASE\_INFO\_S, \*LPNETDEV\_DEV\_CHN\_BASE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| dwDevID | Device IP |
| dwOrgID | Organization ID |
| dwChnType | Channel type. See [NETDEV\_CHN\_TYPE\_E](#_通道类型). |
| dwChnStatus | Channel status. See [NETDEV\_CHN\_STATUS\_E](#_编码通道状态). |
| dwChnIndex | Channel index |
| szChnName | Channel name |
| udwRight | Channel permissions |
| udwAbnormalReason | Cause for video channel exceptions |
| byRes | Reserved field. |

**See also:**

[NETDEV\_DEV\_CHN\_XW\_ENCODE\_INFO\_S](#_设备XW本地信号源通道信息)

### Basic device info

|  |
| --- |
| typedef struct tagNETDEVDeviceBaseInfo  {  INT32 dwChlID;  INT32 dwDeviceType;  INT32 dwDevCode;  INT32 dwCurrentLang;  [NETDEV\_DATE\_INFO\_S](#_时间信息) stBuildDate;  CHAR szSerialNum[NETDEV\_LEN\_32];  CHAR szMacAddress[NETDEV\_LEN\_64];  CHAR szDevName[NETDEV\_CODE\_STR\_MAX\_LEN];  CHAR szDevModel[NETDEV\_CODE\_STR\_MAX\_LEN];  CHAR szVIIDVersion[NETDEV\_CODE\_STR\_MAX\_LEN];  CHAR szSoftwareVersion[NETDEV\_CODE\_STR\_MAX\_LEN];  CHAR szFirmwareVersion[NETDEV\_CODE\_STR\_MAX\_LEN];  CHAR szHardewareID[NETDEV\_CODE\_STR\_MAX\_LEN];  CHAR szUbootVersion[NETDEV\_CODE\_STR\_MAX\_LEN];  CHAR szManufacturer[NETDEV\_LEN\_64];  [NETDEV\_CLIENT\_VERSION\_INFO\_LIST\_S](#_客户端版本信息列表) stClientVersionsInfo;  CHAR szProtoName[NETDEV\_LEN\_256];  BYTE byRes[216];  }NETDEV\_DEVICE\_BASE\_INFO\_S, \*LPNETDEV\_DEVICE\_BASE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChlID | Video input channel ID |
| dwDeviceType | Device type. See [NETDEV\_DEVICE\_TYPE\_E](#_错误码列表). |
| dwDevCode | Device ID |
| dwCurrentLang | Current device language. See [NETDEV\_LANG\_TYPE\_E](#_语言类型枚举). |
| stBuildDate | Build date |
| szSerialNum | Device serial number |
| szMacAddress | MAC address (IPv4) |
| szDevName | Device name |
| szDevModel | Device model |
| szVIIDVersion | VIID component version |
| szSoftwareVersion | Internal software version |
| szFirmwareVersion | Firmware version |
| szHardewareID | Hardware ID |
| szUbootVersion | UBOOT version |
| szManufacturer | Manufacturer name. Length: [1,64] |
| stClientVersionsInfo | Version info. Optional. Applicable to VMS only. |
| szProtoName | Device prototype name |
| byRes | Reserved field. |

**See also:**

[NETDEV\_GetDeviceBaseInfo](#_获取设备基础信息)

### Time info

|  |
| --- |
| typedef struct tagstNETDEVDateInfo  {  INT32 dwYear;  INT32 dwMonth;  INT32 dwDay;  INT32 dwHour;  INT32 dwMinute;  BYTE byRes[32];  }NETDEV\_DATE\_INFO\_S, \*LPNETDEV\_DATE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwYear | Year |
| dwMonth | Month |
| dwDay | Day |
| dwHour | Hour |
| dwMinute | Minute |
| byRes | Reserved field. |

**See also:**

[NETDEV\_DEVICE\_BASE\_INFO\_S](#_设备基础信息)

### Client version list

|  |
| --- |
| typedef struct tagstNETDEVClientVersionsInfoList  {  UINT32 udwNum;  [LPNETDEV\_CLIENT\_VERSION\_INFO\_S](#_客户端版本信息) pstClientVersionsInfo;  BYTE byRes[32];  }NETDEV\_CLIENT\_VERSION\_INFO\_LIST\_S, \*LPNETDEV\_CLIENT\_VERSION\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Quantity |
| pstClientVersionsInfo | Client version list |
| byRes | Reserved field. |

**See also:**

[NETDEV\_DEVICE\_BASE\_INFO\_S](#_设备基础信息)

### Client version info

|  |
| --- |
| typedef struct tagstNETDEVClientVersionsInfo  {  UINT32 udwType;  CHAR szVersion[NETDEV\_LEN\_256];  BYTE byRes[32];  }NETDEV\_CLIENT\_VERSION\_INFO\_S, \*LPNETDEV\_CLIENT\_VERSION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | Client type. See  [NETDEV\_CLIENT\_TYPE\_E](#_客户端类型). |
| szVersion | Client version number. Range: [0, 255] |
| byRes | Reserved field. |

**See also:**

[NETDEV\_CLIENT\_VERSION\_INFO\_LIST\_S](#_客户端版本信息列表)

### Device info

|  |
| --- |
| typedef struct tagNETDEVDeviceInfo  {  INT32 dwDevType;  INT16 wAlarmInPortNum;  INT16 wAlarmOutPortNum;  INT32 dwChannelNum;  BYTE byRes[48];  }NETDEV\_DEVICE\_INFO\_S,\*LPNETDEV\_DEVICE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwDevType | Device type. See [NETDEV\_DEVICE\_TYPE\_E](#_错误码列表). |
| wAlarmInPortNum | Number of alarm inputs |
| wAlarmOutPortNum | Number of alarm outputs |
| dwChannelNum | Number of channels |
| byRes | Reserved field. |

**See also:**

[NETDEV\_GetDeviceInfo](#_获取设备信息)

### Detailed device info

|  |
| --- |
| typedef struct tagstNETDEVDeviceDetailInfo  {  [NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1) stDevBasicInfo;  [NETDEV\_DEV\_FIREWARE\_INFO\_S](#_设备固件信息) stDevFirewareInfo;  }NETDEV\_DEV\_INFO\_V30\_S, \*LPNETDEV\_DEV\_INFO\_V30\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stDevBasicInfo | Basic device info |
| stDevFirewareInfo | Device firmware info |

**See also:**

[NETDEV\_GetDeviceInfo\_V30](#_查询设备详细信息)

### Device firmware info

|  |
| --- |
| typedef struct tagstNETDEVDeviceFirmwareInfo  {  CHAR szDevModel[NETDEV\_CODE\_STR\_MAX\_LEN];  CHAR szFireVersion[NETDEV\_CODE\_STR\_MAX\_LEN];  CHAR szSerialNum[NETDEV\_CODE\_STR\_MAX\_LEN];  BYTE byRes[128];  }NETDEV\_DEV\_FIREWARE\_INFO\_S, \*LPNETDEV\_DEV\_FIREWARE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szDevModel | Device model |
| szFireVersion | Firmware version |
| szSerialNum | Device serial number |
| byRes | Reserved field. |

**See also:**

[NETDEV\_DEV\_INFO\_V30\_S](#_设备详细信息)

### Geolocation info

|  |
| --- |
| typedef struct tagNETDEVGeolocationInfo  {  FLOAT fLongitude;  FLOAT fLatitude;  }NETDEV\_GEOLACATION\_INFO\_S, \*LPNETDEV\_GEOLACATION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| fLongitude | Longitude |
| fLatitude | Latitude |

**See also:**

[NETDEV\_GetGeolocationInfo](#_获取设备定位信息)

### Wi-Fi sniffer MAC array info

|  |
| --- |
| typedef struct tagNETDEVWiFiSnifferMacList  {  INT32 dwSize;  [NETDEV\_WIFISNIFFER\_MAC\_INFO\_S](#_WIFISnifferMac信息) astMACList[NETDEV\_WIFISNIFFER\_MAC\_ARRY\_MAX\_NUM];  }NETDEV\_WIFISNIFFER\_MAC\_LIST\_S, \*LPNETDEV\_WIFISNIFFER\_MAC\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Quantity |
| astMACList | Wi-Fi sniffer MAC info |

**See also:**

[NETDEV\_GetWifiSnifferMacList](#_获取wifi_sniffer_MAC地址信息)

### Wi-Fi sniffer MAC info

|  |
| --- |
| typedef struct tagNETDEVWiFiSnifferMacInfo  {  CHAR aszMACInfo[NETDEV\_WIFISNIFFER\_MAC\_MAX\_NUM];  INT16 wMaxSig;  CHAR aszMaxSigTime[NETDEV\_MAX\_DATE\_STRING\_LEN];  }NETDEV\_WIFISNIFFER\_MAC\_INFO\_S, \*LPNETDEV\_WIFISNIFFER\_MAC\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| aszMACInfo | MAC address of Wi-Fi sniffer |
| wMaxSig | Max signal strength corresponding to the MAC address (unit: dBm) |
| aszMaxSigTime | Acquisition time of max signal strength |

**See also:**

[NETDEV\_WIFISNIFFER\_MAC\_LIST\_S](#_WIFISnifferMac数组信息)

### Detailed video channel info

|  |
| --- |
| typedef struct tagNETDEVVideoChlDetailInfo  {  INT32 dwChannelID;  BOOL bPtzSupported;  [NETDEV\_CHANNEL\_STATUS\_E](#_通道状态) enStatus;  INT32 dwStreamNum;  CHAR szChnName[NETDEV\_LEN\_64];  BYTE byRes[4];  }NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_S, \*LPNETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| bPtzSupported | Whether PTZ is supported. |
| enStatus | Channel status |
| dwStreamNum | Number of streams. The value is invalid when enStatus is [NETDEV\_CHL\_STATUS\_UNBIND](#_通道状态). |
| szChnName | Channel name |
| byRes | Reserved field. |

**See also:**

NETDEV\_QueryVideoChlDetailList

### Detailed video channel info

|  |
| --- |
| typedef struct tagNETDEVVideoChlDetailInfoEx  {  INT32 dwChannelID;  BOOL bPtzSupported;  [NETDEV\_CHANNEL\_STATUS\_E](#_通道状态) enStatus;  INT32 dwStreamNum;  [NETDEV\_CHANNEL\_TYPE\_E](#_通道类型_1) enChannelType;  [NETDEV\_VIDEO\_STANDARD\_E](#_视频制式) enVideoFormat;  [NETDEV\_HOSTTYPE\_E](#_IP协议类型) enAddressType;  CHAR szIPAddr[NETDEV\_IPADDR\_STR\_MAX\_LEN];  INT32 dwPort;  CHAR szChnName[NETDEV\_LEN\_64];  BOOL allowDistribution;  INT32 dwDeviceType;  CHAR szManufacturer[NETDEV\_LEN\_32];  CHAR szDeviceModel[NETDEV\_LEN\_32];  UINT32 udwAccessProtocol;  [LPNETDEV\_VIDEO\_CHL\_DETAIL\_EXTEND\_INFO\_S](#_视频通道详细信息拓展) pstExtendedInformation ;  BYTE byRes[16];  }NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S, \*LPNETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| bPtzSupported | Whether PTZ is supported |
| enStatus | Channel status |
| dwStreamNum | Number of streams |
| enChannelType | Channel type |
| enVideoFormat | Video input format. Invalid when ChannelType is NETDEV\_CHL\_TYPE\_DIGITAL. (note: Effective to hybrid NVR only) |
| enAddressType | IP address type |
| szIPAddr | IP address |
| dwPort | Port number |
| szChnName | Channel name |
| allowDistribution | Whether allows CDN |
| dwDeviceType | Type of connected device. See [NETDEV\_CHANNEL\_CAMERA\_TYPE\_E](#_通道下接入的设备类型(目前仅NVR使用)). |
| szManufacturer | Manufacturer. Range: [0,31] |
| szDeviceModel | Device model. Range: [0,31] |
| udwAccessProtocol | Protocol type. ONVIF: 1 Private protocol: 2 GB/T28181: 3 Custom protocol: Custom protocol. Use [NETDEV\_GetDevConfig](#_获取设备的配置信息)/[NETDEV\_SetDevConfig](#_设置设备的配置信息) to get and configure custom protocol info. |
| pstExtendedInformation | Extended attribute. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_QueryVideoChlDetailListEx](#_查询视频通道信息列表)

### Detailed video channel info (expansion)

|  |
| --- |
| typedef struct tagNETDEVVideoChlDetailExtendInfo  {  UINT32 udwIsPoEPort;  UINT32 udwPoEStatus;  CHAR szAccessAddress[NETDEV\_IPADDR\_STR\_MAX\_LEN];  CHAR szMACAddress[NETDEV\_IPADDR\_STR\_MAX\_LEN];  CHAR szDDNSAddress[NETDEV\_IPADDR\_STR\_MAX\_LEN];  UINT32 udwOffReason;  UINT32 udwRemoteIndex;  CHAR szGBID[NETDEV\_LEN\_32];  UINT32 udwAddType;  UINT32 udwOrgID;  UINT32 udwDevID;  CHAR szDevName[NETDEV\_LEN\_64];  UINT32 udwChlIndex;  UINT32 udwAudioResID;  BYTE byRes[512];  }NETDEV\_VIDEO\_CHL\_DETAIL\_EXTEND\_INFO\_S, \*LPNETDEV\_VIDEO\_CHL\_DETAIL\_EXTEND\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwIsPoEPort | Whether PoE port is supported. 0: No 1: Yes |
| udwPoEStatus | PoE port status. See [NETDEV\_POE\_STATUS\_E](#_POE口状态类型枚举). |
| szAccessAddress | Device access address. Range: [0, 63]. Required for NVR. |
| szMACAddress | Device MAC address. Range: [0, 63]. Applicable to NVR. |
| szDDNSAddress | DDNS server address. Range: [0,64]. |
| udwOffReason | Offline cause. [NETDEV\_CHN\_OFF\_REASON\_E](#_接入设备离线原因类型枚举). |
| udwRemoteIndex | Remote channel ID (channel ID of multi-channel device) |
| szGBID | GB resource ID. Range: [0, 31] |
| udwAddType | Adding mode. 0: Manual 1: Plug and Play |
| udwOrgID | Organization ID. Required for VMS. |
| udwDevID | Parent device ID. Required for VMS. |
| szDevName | Parent device name. Range: [0, 64]. Required for VMS. |
| udwChlIndex | Channel index ID. Required for VMS. |
| udwAudioResID | Corresponding audio channel ID. Required for VMS. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S](#_视频通道详细信息_1)

### Alarm/event report info

|  |
| --- |
| typedef struct tagNETDEVReportInfo  {  INT32 dwReportType;  [NETDEV\_ALARM\_INFO\_V30\_S](#_告警消息) stAlarmInfo;  [NETDEV\_EVENT\_INFO\_S](#_事件信息) stEventInfo;  }NETDEV\_REPORT\_INFO\_S, \*LPNETDEV\_REPORT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwReportType | Report type. See [NETDEV\_REPORT\_TYPE\_E](#_告警/事件上报类型). |
| stAlarmInfo | Alarm info. Valid when dwReportType is [NETDEV\_REPORT\_TYPE\_ALARM](#_告警/事件上报类型). |
| stEventInfo | Event info. Valid when dwReportType is [NETDEV\_REPORT\_TYPE\_EVENT](#_告警/事件上报类型). |

**See also:**

[NETDEV\_SetAlarmCallBack\_V30](#_注册回调函数,接收报警消息等)

### Alarm message

|  |
| --- |
| typedef struct tagNETDEVAlarmInfoV30  {  INT32 dwAlarmType;  INT32 dwAlarmSubType;  INT32 dwAlarmLevel;  INT64 tAlarmTimeStamp;  INT32 dwChannelID;  INT32 dwAlarmID;  INT32 dwAlarmSrcID;  CHAR szAlarmSrc[NETDEV\_ALARM\_SOURCE\_MAX\_LEN];  BOOL IsAlarmSnapExisted;  UINT16 wIndex;  INT32 dwTotalBandWidth;  INT32 dwUnusedBandwidth;  INT32 dwTotalStreamNum;  INT32 dwFreeStreamNum;  INT32 dwMediaMode;  INT32 dwEventCode;  INT32 dwReserved;  CHAR szFileName[NETDEV\_LEN\_256];  CHAR szDeviceID[NETDEV\_LEN\_64];  CHAR szRelatedID[NETDEV\_LEN\_32];  INT32 dwObjectNum;  LPNETDEV\_OBJECT\_LIST\_S pstObjectList;  BYTE byRes[144];  }NETDEV\_ALARM\_INFO\_V30\_S, \*LPNETDEV\_ALARM\_INFO\_V30\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwAlarmType | Alarm type. See [NETDEV\_ALARM\_TYPE\_E](#_告警类型枚举_1). |
| dwAlarmSubType | Alarm sub type. See [NETDEV\_ALARM\_SUBTYPE\_E](#_告警子类型枚举). |
| dwAlarmLevel | Alarm level. 1 to 5 (1 is the severest) |
| tAlarmTimeStamp | Alarm time |
| dwChannelID | Channel ID. Applicable to non-VMS devices. |
| dwAlarmID | Alarm ID. Applicable to VMS. |
| dwAlarmSrcID | Alarm source ID. See [NETDEV\_ALARM\_SRC\_TYPE\_E](#_告警资源类型). |
| szAlarmSrc | Alarm source info (name) |
| IsAlarmSnapExisted | Whether an alarm snapshot exists. 0: No 1: Yes. Applicable to VMS. |
| wIndex | Index number |
| dwTotalBandWidth | Current total bandwidth (unit: Mbps) |
| dwUnusedBandwidth | Current idle bandwidth (unit: Mbps) |
| dwTotalStreamNum | Total number of streams |
| dwFreeStreamNum | Number of unused streams |
| dwMediaMode | Stream type. See [NETDEV\_MEDIA\_MODE\_E](#_媒体类型枚举类型定义). |
| dwEventCode | Event type. Used to report decoding layer events. See  [NETDEV\_PLAYER\_RUN\_INFO\_TYPE\_E](#_解码层上报运行信息的类型的枚举定义) |
| dwReserved | Reserved parameters for reporting decoding layer events. |
| szFileName | Parameter for reporting strings. |
| szDeviceID | Alarm device ID. Filled in with the device’s GB code if the device is connected via GB. Length: [1,32]. Applicable to IPC and VM. |
| szRelatedID | ID that relates an alarm event with alarm data. Must be unique in a camera. 15 characters long. Required if alarm-related data exists. |
| dwObjectNum | Number of objects |
| pstObjectList | Object list. Memory needs to be allocated dynamically according to udwSize. See [NETDEV\_OBJECT\_LIST\_S](#_目标列表结构题). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_REPORT\_INFO\_S](#_告警/事件上报信息)

### Event info

|  |
| --- |
| typedef struct tagNETDEVEventInfo  {  INT32 dwSize;  [NETDEV\_EVENT\_RES\_S](#_事件资源信息) astEventRes[NETDEV\_MAX\_EVENT\_RES\_SIZE];  INT32 dwEventActionType;  [LPNETDEV\_EVENT\_RES\_S](#_事件资源信息) pstEventRes;  BYTE byRes[28];  }NETDEV\_EVENT\_INFO\_S, \*LPNETDEV\_EVENT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Number of resources |
| astEventRes | Event resource info |
| dwEventActionType | Event type. See [NETDEV\_EVENT\_ACTION\_TYPE\_E](#_事件动作类型枚举). |
| pstEventRes | Memory needs to be allocated dynamically if the event exceeds NETDEV\_MAX\_EVENT\_RES\_SIZE. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_REPORT\_INFO\_S](#_告警/事件上报信息)

### Event resource info

|  |
| --- |
| typedef struct tagNETDEVEventRes  {  INT32 dwResType;  INT32 dwResID;  INT32 dwFirstSubResID;  INT32 dwSecondSubResID;  BYTE byRes[16];  }NETDEV\_EVENT\_RES\_S, \*LPNETDEV\_EVENT\_RES\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwResType | Resource type. See [NETDEV\_EVENT\_RES\_TYPE\_E](#_事件资源类型枚举). |
| dwResID | Resource ID |
| dwFirstSubResID | First sub resource ID |
| dwSecondSubResID | Second sub resource ID. Different resource types have different meanings. For example, the resource ID of video wall window resources is video wall ID, the first sub resource ID is the window ID, and the second sub resource ID is the split window ID. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_EVENT\_INFO\_S](#_事件信息)

### Pull alarm info

|  |
| --- |
| typedef struct tagNETDEVPullAlarmInfo  {  [NETDEV\_ALARM\_TYPE\_E](#_告警类型枚举_1) enAlarm;  INT32 dwChancelId;  }NETDEV\_PULLALARM\_INFO\_S, \*LPNETDEV\_PULLALARM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| enAlarm | Alarm type |
| dwChancelId | Channel ID, applicable to NVR |

**See also:**

NETDEV\_PullAlarm

### People counting

|  |
| --- |
| typedef struct tagNETDEVTrafficStatisticsInfo  {  BOOL bIsSuccess; /\* Query if succeeded\*/  INT32 dwChannelID; /\* Channel ID\*/  UINT32 udwSize; /\* Number of time periods\*/  UINT32 audwEnterCount[NETDEV\_PEOPLE\_CNT\_MAX\_NUM]; /\* Number of people entered\*/  UINT32 audwExitCount[NETDEV\_PEOPLE\_CNT\_MAX\_NUM]; /\* Number of people left \*/  BYTE byRes[6]; /\* Reserved field \*/  }NETDEV\_TRAFFIC\_STATISTICS\_INFO\_S, \*LPNETDEV\_TRAFFIC\_STATISTICS\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bIsSuccess | Query whether succeeded |
| dwChannelID | Channel ID |
| udwSize | Number of time periods |
| audwEnterCount | Count the number of people that have entered |
| audwExitCount | Count the number of people that have left. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_FindNextTrafficStatisticInfo](#_逐个查询获取到的客流量信息)

### People counting command

|  |
| --- |
| typedef struct tagNETDEVTrafficStatisticsCond  {  INT32 dwChannelID;  INT32 dwStatisticsType;  INT32 dwFormType;  INT64 tBeginTime;  INT64 tEndTime;  }NETDEV\_TRAFFIC\_STATISTICS\_COND\_S, \*LPNETDEV\_TRAFFIC\_STATISTICS\_COND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| dwStatisticsType | Statistics type. See [NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_E](#_客流量统计模式类型) |
| dwFormType | Statistical report. See [NETDEV\_FORM\_TYPE\_E](#_客流量统计报表类型) |
| tBeginTime | Start time |
| tEndTime | End time |

**See also:**

[NETDEV\_GetTrafficStatistic](#_获取客流量统计), [NETDEV\_StartTrafficStatistic](#_开始客流量统计查询)

### People counting

|  |
| --- |
| typedef struct tagNETDEVTrafficStatisticsData  {  INT32 dwSize;  INT32 adwEnterCount[NETDEV\_PEOPLE\_CNT\_MAX\_NUM];  INT32 adwExitCount[NETDEV\_PEOPLE\_CNT\_MAX\_NUM];  }NETDEV\_TRAFFIC\_STATISTICS\_DATA\_S, \*LPNETDEV\_TRAFFIC\_STATISTICS\_DATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Report length. Varies with report type. |
| adwEnterCount | Count of people that have entered |
| adwExitCount | Count of people that have left |

**See also:**

[NETDEV\_GetTrafficStatistic](#_获取客流量统计)

### Time interval for reporting people counting statistics

|  |
| --- |
| typedef struct tagNETDEVTrafficStatisticsReportTime  {  UINT32 udwReportTimeInterval;  BOOL bEnable;  }NETDEV\_TRAFFIC\_STATISTICS\_REPORT\_TIME\_S, \*LPNETDEV\_TRAFFIC\_STATISTICS\_REPORT\_TIME\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwReportTimeInterval | Time interval (unit: second) |
| bEnable | Enable or disable. 0: Disable 1: Enable |

**See also:**

[NETDEV\_GetTrafficStatisticReportTime](#_获取客流量上报时间间隔)

### Multi-channel people counting command

|  |
| --- |
| typedef struct tagNETDEVMultiTrafficStatisticsCond  {  [NETDEV\_OPERATE\_LIST\_S](#_批处理列表) stChannelIDs;  UINT32 udwStatisticsType;  UINT32 udwFormType;  INT64 tBeginTime;  INT64 tEndTime;  BYTE byRes[256];  }NETDEV\_MULTI\_TRAFFIC\_STATISTICS\_COND\_S, \*LPNETDEV\_MULTI\_TRAFFIC\_STATISTICS\_COND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stChannelIDs | Channel ID list |
| udwStatisticsType | See [NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_E](#_客流量统计模式类型). |
| udwFormType | See [NETDEV\_TRAFFIC\_STATIC\_FORM\_TYPE\_E](#_客流量报表类型). |
| tBeginTime | Start time |
| tEndTime | End time |
| byRes | Reserved field |

**See also:**

[NETDEV\_StartMultiTrafficStatistic](#_多通道开始客流量统计查询)

### People counting info

|  |
| --- |
| typedef struct tagNETDEVPassengerFlowStatisticData  {  INT32 dwChannelID;  INT64 tReportTime;  INT32 tInterval;  INT32 dwEnterNum;  INT32 dwExitNum;  INT32 dwTotalEnterNum;  INT32 dwTotalExitNum;  }NETDEV\_PASSENGER\_FLOW\_STATISTIC\_DATA\_S, \*LPNETDEV\_PASSENGER\_FLOW\_STATISTIC\_DATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| tReportTime | Report time (Unix timestamp) |
| tInterval | Time interval |
| dwEnterNum | Number of people that have entered |
| dwExitNum | Number of people that have left |
| dwTotalEnterNum | Total number of people that have entered |
| dwTotalExitNum | Total number of people that have left |

**See also:**

[NETDEV\_PassengerFlowStatisticCallBack\_PF](#_注册客流量统计上报回调函数)

### Structure of info about Xware device capability

|  |
| --- |
| typedef struct tagNETDEVXWCAPINFO  {  [NETDEV\_XW\_TVWALL\_CAP\_S](#_电视墙能力集结构体) stTVWall;  [NETDEV\_XW\_FORMAT\_LIST\_S](#_视频输出制式列表结构体) stVideoFormats;  [NETDEV\_XW\_Wnd\_CAP\_S](#_窗口能力集结构体) stWnd;  [NETDEV\_XW\_BASEMAP\_CAP\_S](#_底图能力集结构体) stBaseMap;  [NETDEV\_XW\_VIRTUALLED\_CAP\_S](#_虚拟LED能力集结构体) stVirtualLED;  [NETDEV\_XW\_SEQUENCE\_CAP\_S](#_轮巡能力集结构体) stSequence;  [NETDEV\_XW\_SUPPORT\_CAP\_S](#_特性能力集结构体) stSupport;  BOOL bIsRSASupported;  BOOL bIsDevMgrSupported;  BOOL bIsDecodeSupported;  BOOL bIsChromaCalSupported;  UINT32 udwSuggestVideoFormat;  UINT32 udwSupportExVideoFormatNum;  [LPNETDEV\_XW\_FORMAT\_EX\_LIST\_S](#_"XW"设备扩展视频输出制式能力集结构体) pstVideoFormatsEx;  UINT32 udwBatchCapabilities;  BOOL bIsOrderNOSupported;  LPNETDEV\_XW\_ALARM\_CAP\_S pstAlarmCap;  UINT32 udwDeviceID;  BYTE byRes[24];  }NETDEV\_XW\_CAP\_INFO\_S, \*LPNETDEV\_XW\_CAP\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stTVWall | Video wall capability |
| stVideoFormats | Video output format list |
| stWnd | Window capability |
| stBaseMap | Background image capability |
| stVirtualLED | Virtual LED capability |
| stSequence | Sequence display capability |
| stSupport | Capability of features |
| bIsRSASupported | Whether RSA is supported |
| bIsDevMgrSupported | Whether device management is supported |
| bIsDecodeSupported | Whether decoding is supported |
| bIsChromaCalSupported | Whether chroma correction is supported |
| udwSuggestVideoFormat | Recommended resolution. See [NETDEV\_VIDEO\_FORMAT\_E](#_分辨率枚举). |
| udwSupportExVideoFormatNum | Number of supported expansion video output formats |
| pstVideoFormatsEx | List of supported expansion video output formats. Memory needs to be allocated dynamically according to udwSupportExVideoFormatNum. |
| udwBatchCapabilities | Batch operation capability. Described by bit. Bit1: Open/close window Bit2: Roam Bit3: Split window Bit4: Window ID. |
| bIsOrderNOSupported | Whether sequence number is supported. |
| pstAlarmCap | Device alarm capability |
| udwDeviceID | Device ID. Applicable to VMS only. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_GetCapability](#_获取能力集)

### Structure of video wall capability

|  |
| --- |
| typedef struct tagNETDEVXWTVWallCap  {  UINT32 udwNumMax;  UINT32 udwSceneMax;  UINT32 udwRowMax;  UINT32 udwColumnMax;  UINT32 udwScenesPlanMax;  UINT32 udwMaxPerTVWallVideoOutNum;  BYTE byRes[56];  }NETDEV\_XW\_TVWALL\_CAP\_S, \*LPNETDEV\_XW\_TVWALL\_CAP\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNumMax | Maximum number of video walls |
| udwSceneMax | Maximum number of scenes per video wall |
| udwRowMax | Maximum number of rows per video wall |
| udwColumnMax | Maximum number of rows per video wall |
| udwScenesPlanMax | Maximum number of scene group plans per video wall |
| udwMaxPerTVWallVideoOutNum | Maximum number of video outputs per video wall. The product of the rows and columns of the video wall cannot exceed the limit. |
| byRes | Reserved field. |

### Structure of video output format list

|  |
| --- |
| typedef struct tagNETDEVXWFormatList  {  UINT32 udwSize;  UINT32 adwFormatList[NETDEV\_VIDEO\_FORMAT\_MAX];  BYTE byRes[64];  }NETDEV\_XW\_FORMAT\_LIST\_S, \*LPNETDEV\_XW\_FORMAT\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSize | Number of video output formats supported |
| adwFormatList | Supported video output formats. See [NETDEV\_VIDEO\_FORMAT\_E](#_分辨率枚举). |
| byRes | Reserved field. |

### Structure of window capability

|  |
| --- |
| typedef struct tagNETDEVXWWndCap  {  UINT32 udwWndMaxPerScene;  UINT32 udwAreaMax;  UINT32 udwPaneMax;  [NETDEV\_XW\_RESOLUTION\_S](#_分辨率信息) stResolutionMin;  BYTE byRes[64];  }NETDEV\_XW\_Wnd\_CAP\_S, \*LPNETDEV\_XW\_Wnd\_CAP\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwWndMaxPerScene | Maximum number of windows per scene |
| udwAreaMax | Maximum window size per physical output |
| udwPaneMax | Maximum number of split windows per window |
| stResolutionMin | Minimum window resolution |
| byRes | Reserved field. |

### Structure of background image capability

|  |
| --- |
| typedef struct tagNETDEVXWBaseMapCap  {  UINT32 udwNumMax;  UINT32 udwNumMaxPerScene;  UINT32 udwMaxBaseMapSize;  [NETDEV\_XW\_RESOLUTION\_S](#_分辨率信息) stResolutionMax;  [NETDEV\_XW\_RESOLUTION\_S](#_分辨率信息) stResolutionMin;  CHAR szBaseMapType[NETDEV\_LEN\_40];  BYTE byRes[16];  }NETDEV\_XW\_BASEMAP\_CAP\_S, \*LPNETDEV\_XW\_BASEMAP\_CAP\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNumMax | Maximum number of background images supported |
| udwNumMaxPerScene | Maximum number of background images per scene |
| udwMaxBaseMapSize | Supported image size (unit: MB) |
| stResolutionMax | Maximum resolution of background image |
| stResolutionMin | Minimum resolution of background image |
| szBaseMapType | Supported image format |
| byRes | Reserved field. |

### Structure of virtual LED capability

|  |
| --- |
| typedef struct tagNETDEVXWVirtualLEDCap  {  UINT32 udwNumMaxPerScene;  UINT32 udwRowMax;  UINT32 udwCharMaxPerRow;  UINT32 udwRollingNumPerSceneMax;  UINT32 udwTextHightMax;  UINT32 udwGapMax;  UINT32 udwCharsMax;  UINT32 udwSupVirtualLEDDifColors;  UINT32 udwMaxPerSceneTimeLEDNum;  BYTE byRes[40];  }NETDEV\_XW\_VIRTUALLED\_CAP\_S, \*LPNETDEV\_XW\_VIRTUALLED\_CAP\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNumMaxPerScene | Maximum number of virtual LEDs per scene |
| udwRowMax | Maximum number of rows per virtual LED |
| udwCharMaxPerRow | Maximum number of characters per row |
| udwRollingNumPerSceneMax | Maximum number of scrolling virtual LEDs |
| udwTextHightMax | Maximum font height |
| udwGapMax | Maximum number of pixels allowed for adaptive gap |
| udwCharsMax | Maximum number of characters |
| udwSupVirtualLEDDifColors | Whether virtual LED can display different colors. 0: No 1: Yes |
| udwMaxPerSceneTimeLEDNum | Maximum number of time virtual LEDs per scene |
| byRes | Reserved field. |

### Structure of sequence display capability

|  |
| --- |
| typedef struct tagNETDEVXWSequenceCap  {  UINT32 udwSrcNumMax;  UINT32 udwPaneNumMax;  UINT32 udwWndNumMax;  UINT32 udwSceneNumMax;  UINT32 udwVideoSourceMax;  UINT32 udwSeqResourceInSceneMax;  BYTE byRes[56];  }NETDEV\_XW\_SEQUENCE\_CAP\_S, \*LPNETDEV\_XW\_SEQUENCE\_CAP\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSrcNumMax | Maximum number of resources per sequence |
| udwPaneNumMax | Maximum number of sequences per split window |
| udwWndNumMax | Maximum number of sequences per window |
| udwSceneNumMax | Maximum number of scene sequences |
| udwVideoSourceMax | Maximum number of cameras per sequence resource |
| udwSeqResourceInSceneMax | Maximum number of sequence resources per scene |
| byRes | Reserved field. |

### Structure of feature capability

|  |
| --- |
| typedef struct tagNETDEVXWSupportCap  {  UINT32 udwPane;  UINT32 udwWndRoaming;  UINT32 udwWnd;  BOOL bBackToShow;  BOOL bWndOverlay;  BOOL bTopWndSetTransp;  UINT32 udwMaxWinNumPerScreen;  [NETDEV\_XW\_PANE\_TYPE\_S](#_XW设备支持的分屏信息结构体) stPaneType;  UINT32 udwMaxEncodeDevicesNum;  UINT32 udwMaxVideoSourceNum;  UINT32 udwSupportWindowTypeNum;  UINT32 \*paudwWindowType;  BOOL bAutoSelectSignalSrc;  BOOL bSupportCustomizeOutputFormat;  BOOL bSupportCustomizeInputFormat;  BYTE byRes[36];  }NETDEV\_XW\_SUPPORT\_CAP\_S, \*LPNETDEV\_XW\_SUPPORT\_CAP\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPane | Split window capability. See [NETDEV\_PANE\_CAP\_E](#_分屏能力枚举) |
| udwWndRoaming | Window roaming capability. See [NETDEV\_ROAMING\_CAP\_E](#_窗口漫游能力枚举) |
| udwWnd | Window opening capability. See [NETDEV\_XW\_WND\_CAP\_E](#_开窗能力枚举) |
| bBackToShow | Whether back display is supported. 0: No 1: Yes |
| bWndOverlay | Whether window overlay is supported. 0: No 1: Yes |
| bTopWndSetTransp | Whether the top-level window supports transparency. 0: No 1: Yes |
| udwMaxWinNumPerScreen | Maximum number of windows per physical output |
| stPaneType | Supported split windows |
| udwMaxEncodeDevicesNum | Maximum number of encoding devices |
| udwMaxVideoSourceNum | Maximum number of video input channels |
| udwSupportWindowTypeNum | Number of window opening modes supported |
| paudwWindowType | Window opening modes. See [NETDEV\_XW\_LAYOUT\_NUM\_E](#_屏幕输出制式枚举). Memory needs to be allocated dynamically according to the number of supported modes. |
| bAutoSelectSignalSrc | Whether auto-selection of signal source is supported. 0: No 1: Yes |
| bSupportCustomizeOutputFormat | Whether custom output format is supported. 0: No 1: Yes |
| bSupportCustomizeInputFormat | Whether custom local input EDID is supported. 0: No 1: Yes |
| byRes | Reserved field. |

### Structure of info about window layout supported by Xware device

|  |
| --- |
| typedef struct tagNETDEVXWPaneType  {  UINT32 udwSize;  UINT32 \*pudwPaneType;  }NETDEV\_XW\_PANE\_TYPE\_S, \*LPNETDEV\_XW\_PANE\_TYPE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSize | Number of split windows supported |
| pudwPaneType | Supported window layouts. See [NETDEV\_XW\_LAYOUT\_NUM\_E](#_屏幕输出制式枚举) |

### Structure of video output expansion format of Xware device

|  |
| --- |
| typedef struct tagNETDEVXWFormatListEx  {  UINT32 udwVideoOutID;  UINT32 udwFormatNum;  UINT32 adwFormatList[NETDEV\_VIDEO\_FORMAT\_MAX];  BYTE byRes[64];  }NETDEV\_XW\_FORMAT\_EX\_LIST\_S, \*LPNETDEV\_XW\_FORMAT\_EX\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwVideoOutID | Video output channel ID, which is the unique identifier of a video output channel. Obtained through the interface for getting device’s physical resources. Same as the VideoOutID passed by the NETDEV\_XW\_CreateTVWallCfg and NETDEV\_XW\_CreateTVWallCfg\_V30 interface. |
| udwFormatNum | Number of video formats supported by video output channel |
| adwFormatList | Supported video output formats. See [NETDEV\_VIDEO\_FORMAT\_E](#_分辨率枚举). |
| byRes | Reserved field. |

### Structure of info about the number of channels

|  |
| --- |
| typedef struct tagNETDEVXWChannelsNum  {  UINT32 udwVideoInNum;  UINT32 udwVideoOutNum;  UINT32 udwAudioInNum;  UINT32 udwAudioOutNum;  BYTE byRes[64];  }NETDEV\_XW\_CHANNELS\_NUM\_S, \*LPNETDEV\_XW\_CHANNELS\_NUM\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwVideoInNum | Number of video input channels |
| udwVideoOutNum | Number of video output channels |
| udwAudioInNum | Number of audio input channels |
| udwAudioOutNum | Number of audio output channels |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_GetChannelsNum](#_获取本地编码通道数量)

### Structure of info about decoding channels

|  |
| --- |
| typedef struct tagNETDEVXWChannelsList  {  UINT32 udwVideoInNum;  [LPNETDEV\_XW\_VIDEO\_IN\_INFO\_S](#_视频输入通道信息结构体) pstVideoIn;  UINT32 udwVideoOutNum;  [LPNETDEV\_XW\_VIDEO\_OUT\_INFO\_S](#_XW视频输出通道信息结构体) pstVideoOut;  UINT32 udwAudioInNum;  [LPNETDEV\_XW\_AUDIO\_IN\_INFO\_S](#_音频输入通道信息结构体) pstAudioIn;  UINT32 udwAudioOutNum;  [LPNETDEV\_XW\_AUDIO\_OUT\_INFO\_S](#_音频输出通道信息结构体) pstAudioOut;  UINT32 udwSerialNum;  [LPNETDEV\_XW\_SERIAL\_INFO\_S](#_串口通道信息结构体) pstSerialInfo;  BYTE byRes[56];  }NETDEV\_XW\_CHANNELS\_LIST\_S, \*LPNETDEV\_XW\_CHANNELS\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwVideoInNum | Number of video input channels. The maximum number of channels supported by ADU/CDU is required. |
| pstVideoIn | Video input channel info. Memory needs to be allocated dynamically according to udwVideoInNum. |
| udwVideoOutNum | Number of video output channels. The maximum number of channels supported by ADU/CDU is required. |
| pstVideoOut | Video output channel info. Memory needs to be allocated dynamically according to udwVideoOutNum. |
| udwAudioInNum | Number of audio input channels. The maximum number of channels supported by ADU/CDU is required. |
| pstAudioIn | Audio input channel info. Memory needs to be allocated dynamically according to udwAudioInNum. |
| udwAudioOutNum | Number of audio output channels. The maximum number of channels supported by ADU/CDU is required. |
| pstAudioOut | Audio output channel info. Memory needs to be allocated dynamically according to udwAudioOutNum. |
| udwSerialNum | Number of serial port channels. The maximum number of channels supported by ADU/CDU is required. |
| pstSerialInfo | Serial port channel info. Memory needs to be allocated dynamically according to udwSerialNum. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_GetChannelsInfoList](#_获取视频通道信息列表)

### Structure of info about video input channels

|  |
| --- |
| typedef struct tagNETDEVXWVideoInInfo  {  INT32 dwChannelID;  INT32 dwIndex;  UINT32 udwVideoPortType;  UINT32 udwStatus;  BYTE byRes[60];  }NETDEV\_XW\_VIDEO\_IN\_INFO\_S, \*LPNETDEV\_XW\_VIDEO\_IN\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| dwIndex | Sequence number |
| udwVideoPortType | Channel type. See [NETDEV\_VIDEO\_PORT\_TYPE\_E](#_视频端口枚举). |
| udwStatus | Channel status. See [NETDEV\_CHN\_STATUS\_E](#_编码通道状态). |
| byRes | Reserved field. |

### Structure of info about video output channel of Xware device

|  |
| --- |
| typedef struct tagNETDEVXWVideoOutInfo  {  INT32 dwChannelID;  INT32 dwSlotID;  INT32 dwIndex;  UINT32 udwVideoPortType;  UINT32 udwStatus;  }NETDEV\_XW\_VIDEO\_OUT\_INFO\_S, \*LPNETDEV\_XW\_VIDEO\_OUT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| dwSlotID | Slot ID |
| dwIndex | Sequence number |
| udwVideoPortType | Channel type. See [NETDEV\_VIDEO\_PORT\_TYPE\_E](#_视频端口枚举). |
| udwStatus | Channel status. See [NETDEV\_CHN\_STATUS\_E](#_编码通道状态). |

### Structure of info about audio input channel

|  |
| --- |
| typedef struct tagNETDEVXWAudioInInfo  {  INT32 dwChannelID;  UINT32 udwAudioPortType;  UINT32 udwIndex;  UINT32 udwStatus;  BYTE byRes[56];  }NETDEV\_XW\_AUDIO\_IN\_INFO\_S, \*LPNETDEV\_XW\_AUDIO\_IN\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| udwAudioPortType | Channel type. See [NETDEV\_AUDIO\_PORT\_TYPE\_E](#_音频通道端口枚举). |
| udwIndex | Serial number. Same as that printed on the device. |
| udwStatus | Channel status. See [NETDEV\_CHN\_STATUS\_E](#_编码通道状态). |
| byRes | Reserved field. |

### Structure of info about audio output channel

|  |
| --- |
| typedef struct tagNETDEVXWAudioOutInfo  {  INT32 dwChannelID;  UINT32 udwAudioPortType;  UINT32 udwIndex;  UINT32 udwStatus;  BYTE byRes[56];  }NETDEV\_XW\_AUDIO\_OUT\_INFO\_S, \*LPNETDEV\_XW\_AUDIO\_OUT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| udwAudioPortType | Channel type. See [NETDEV\_AUDIO\_PORT\_TYPE\_E](#_音频通道端口枚举). |
| udwIndex | Serial number. Same as that printed on the device. |
| udwStatus | Channel status. See [NETDEV\_CHN\_STATUS\_E](#_编码通道状态). |
| byRes | Reserved field. |

### Structure of info about serial interface channel

|  |
| --- |
| typedef struct tagNETDEVXWSerialInfo  {  INT32 dwChannelID;  INT32 dwIndex;  UINT32 udwSerialPortType;  UINT32 udwStatus;  BYTE byRes[60];  }NETDEV\_XW\_SERIAL\_INFO\_S, \*LPNETDEV\_XW\_SERIAL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| dwIndex | Sequence number |
| udwSerialPortType | Channel type. See [NETDEV\_XW\_SERIAL\_TYPE\_E](#_串口类型枚举). |
| udwStatus | Channel status. See [NETDEV\_CHN\_STATUS\_E](#_编码通道状态). |
| byRes | Reserved field. |

### Structure of video wall info

|  |
| --- |
| typedef struct tagNETDEVXWTVWallCfg  {  UINT32 udwTVWallID;  CHAR szTVWallCode[NETDEV\_LEN\_64];  CHAR szName[NETDEV\_TVWALL\_NAME\_LEN];  UINT32 udwOrderNo;  [NETDEV\_TVWALL\_SCREEN\_S](#_电视墙屏幕配置结构体) stScreenInfo;  UINT32 udwFormatSpecNum;  [NETDEV\_TVWALL\_FORMAT\_SPEC\_S](#_特殊输出制式配置结构体) astFormatSpec[NETDEV\_FORMAT\_SPEC\_MAX];  UINT32 udwLedSpecNum;  [NETDEV\_TVWALL\_LED\_SPEC\_S](#_特殊模组框配置结构体) astLedSpec[NETDEV\_LED\_SPEC\_MAX];  UINT32 udwVideoOutNum;  [NETDEV\_TVWALL\_VIDED\_OUT\_S](#_物理输出端口配置结构体) astVideoOut[NETDEV\_VIDEO\_OUT\_MAX];  UINT32 udwTVWallType;  INT32 dwXWDeviceID;  UINT32 udwZoomEnabled;  BYTE byRes[176];  }NETDEV\_XW\_TVWALL\_CFG\_S, \*LPNETDEV\_XW\_TVWALL\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTVWallID | Video wall ID |
| szTVWallCode | Video wall code (optional). Assigned by device. String length range: [1,64] |
| szName | Video wall name |
| udwOrderNo | Video wall serial number |
| stScreenInfo | Screen configuration |
| udwFormatSpecNum | Number of special output formats |
| astFormatSpec | Configuration of special output formats |
| udwLedSpecNum | Number of LED screens |
| astLedSpec | Configuration of LED screens |
| udwVideoOutNum | Number of physical outputs |
| astVideoOut | Physical output port list |
| udwTVWallType | Video wall type |
| dwXWDeviceID | -1: DC device >0: ADU device ID |
| udwZoomEnabled | Whether zoom is enabled. 0: No 1: Yes |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_CreateTVWallCfg](#_创建电视墙配置), [NETDEV\_XW\_ModifyTVWallCfg](#_修改电视墙配置), [NETDEV\_XW\_FindNextTVWallCfg](#_逐个获取查找到的电视墙配置信息), [NETDEV\_XW\_GetTVWallCfg](#_获取单个电视墙布局配置信息)

### Structure of video wall screen configuration

|  |
| --- |
| typedef struct tagNETDEVTVWallScreen  {  UINT32 udwRowNum;  UINT32 udwColNum;  UINT32 udwFormat;  CHAR szFormat[NETDEV\_FORMAT\_NAME\_LEN];  [NETDEV\_XW\_RESOLUTION\_S](#_分辨率信息) stResolution;  BYTE byRes[64];  }NETDEV\_TVWALL\_SCREEN\_S, \*LPNETDEV\_TVWALL\_SCREEN\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwRowNum | Number of rows on video wall |
| udwColNum | Number of columns on video wall |
| udwFormat | Screen output format. See [NETDEV\_VIDEO\_FORMAT\_E](#_分辨率枚举). |
| szFormat | No configuration needed. Screen output format. |
| stResolution | Screen resolution |
| byRes | Reserved field. |

### Structure of special output format specs

|  |
| --- |
| typedef struct tagNETDEVTVWallFORMATSPECList  {  UINT32 udwScreenID;  UINT32 udwFormat;  CHAR szFormat[NETDEV\_FORMAT\_NAME\_LEN];  BYTE byRes[32];  }NETDEV\_TVWALL\_FORMAT\_SPEC\_S, \*LPNETDEV\_TVWALL\_FORMAT\_SPEC\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwScreenID | Screen ID (starts from 1) |
| udwFormat | Screen output format. See [NETDEV\_VIDEO\_FORMAT\_E](#_分辨率枚举). |
| szFormat | No configuration needed. Screen output format. |
| byRes | Reserved field. |

### Structure of LED screens

|  |
| --- |
| typedef struct tagNETDEVTVWallLEDSPECList  {  UINT32 udwScreenID;  [NETDEV\_XW\_RESOLUTION\_S](#_分辨率信息) stResolution;  BYTE byRes[32];  }NETDEV\_TVWALL\_LED\_SPEC\_S, \*LPNETDEV\_TVWALL\_LED\_SPEC\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwScreenID | Screen ID (starts from 1) |
| stResolution | Screen resolution |
| byRes | Reserved field. |

### Structure of physical output configuration

|  |
| --- |
| typedef struct tagNETDEVTVWallVOUTList  {  UINT32 udwScreenID;  UINT32 udwVideoOutID;  BYTE byRes[32];  }NETDEV\_TVWALL\_VIDED\_OUT\_S, \*LPNETDEV\_TVWALL\_VIDED\_OUT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwScreenID | Screen ID (starts from 1) |
| udwVideoOutID | Physical output port ID |
| byRes | Reserved field. |

### Structure of virtual LED info

|  |
| --- |
| typedef struct tagNETDEVXWVirtualLEDInfo  {  UINT32 udwLEDID;  UINT32 udwModifyCmd;  UINT32 udwEnable;  UINT32 udwType;  [NETDEV\_XW\_AREA\_S](#_相对于电视墙的位置信息) stArea;  [NETDEV\_XW\_BACKGROUND\_S](#_背景结构体) stBackground;  [NETDEV\_XW\_FONT\_INFO\_S](#_文字信息描述结构体) stFontInfo;  CHAR szText[NETDEV\_TEXT\_LEN];  BYTE byRes[32];  }NETDEV\_XW\_VIRTUAL\_LED\_INFO\_S, \*LPNETDEV\_XW\_VIRTUAL\_LED\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwLEDID | Virtual LED ID |
| udwModifyCmd | Modification command word. See [NETDEV\_XW\_LED\_CMD\_E](#_虚拟LED修改可选命令字枚举). |
| udwEnable | Enable or disable virtual LED. 0: Disable 1: Enable |
| udwType | Virtual LED type. See [NETDEV\_XW\_LED\_TYPE\_E](#_虚拟LED类型枚举). |
| stArea | Virtual LED location info |
| stBackground | Background |
| stFontInfo | Font info |
| szText | Text content |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_CreateVirtualLED](#_创建单个虚拟LED), [NETDEV\_XW\_SetVirtualLED](#_配置单个虚拟LED),

[NETDEV\_XW\_GetVirtualLED](#_获取单个虚拟LED), [NETDEV\_XW\_FindNextVirtualLED](#_逐个获取虚拟LED信息)

### Structure of background

|  |
| --- |
| typedef struct tagNETDEVXWBackground  {  UINT32 udwTransparency;  UINT32 udwBaseColor;  BYTE byRes[16];  }NETDEV\_XW\_BACKGROUND\_S, \*LPNETDEV\_XW\_BACKGROUND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTransparency | Transparency. 0: Fully transparent (default) 100: Opaque |
| udwBaseColor | Color. RGB:0xABCDEF AB means R 00~FF CD means G 00~FF EF means B 00~FF default: 0xFFFFFF (white) |
| byRes | Reserved field. |

### Structure of font info

|  |
| --- |
| typedef struct tagNETDEVXWFontInfo  {  UINT32 udwFont;  UINT32 udwSize;  UINT32 udwReferFontSize;  UINT32 udwGap;  UINT32 udwHorizontalAlign;  UINT32 udwVerticalAlign;  UINT32 udwColor;  UINT32 udwScrollMode;  UINT32 udwScrollRateSec;  UINT32 udwReferFontGap;  BOOL bVoiceBroadcast;  BOOL bBold;  BOOL bSlope;  BOOL bUnderline;  UINT32 udwLineSpace;  BYTE byRes[12];  }NETDEV\_XW\_FONT\_INFO\_S, \*LPNETDEV\_XW\_FONT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwFont | Font. See [NETDEV\_XW\_LED\_FONT\_FORMAT\_E](#_虚拟LED字体枚举). |
| udwSize | Font size. For range 0 to 7, see [NETDEV\_XW\_LED\_FONT\_SIZE\_E](#_虚拟LED字体大小枚举). The range for the rest is [48,1024], indicating the actual height (in pixel). |
| udwReferFontSize | Reference font size. Required when udwSize is set to “adaptive”. For the value range, see the Size filed. |
| udwGap | Font space. See [NETDEV\_XW\_LED\_FONT\_GAP\_E](#_虚拟LED字间距枚举). |
| udwHorizontalAlign | Horizontal alignment mode. See [NETDEV\_XW\_HORIZONTAL\_ALIGN\_MODE\_E](#_文字水平对齐方式枚举). |
| udwVerticalAlign | Vertical alignment mode. See [NETDEV\_XW\_VERTICAL\_ALIGN\_MODE\_E](#_文字垂直对齐方式枚举). |
| udwColor | Color. RGB:0xABCDEF AB means R 00~FF CD means G 00~FF EF means B 00~FF default: 0xFFFFFF (white) |
| udwScrollMode | Text scrolling mode. See [NETDEV\_XW\_LED\_SCROLLING\_MODE\_E](#_文字滚动方式枚举). |
| udwScrollRateSec | Scrolling speed (unit: second). Required when ScrollingMode is not 0. Range: 1x-10x (1x is the default and the slowest). |
| udwReferFontGap | Reference font spacing. Valid when udwGap is “adaptive”. Range: 0-1024 (actual width in pixel) |
| bVoiceBroadcast | Voice broadcast 0: No 1: Yes |
| bBold | Bold. 0: No 1: Yes |
| bSlope | Slope. 0: No 1: Yes |
| bUnderline | Underline. 0: No 1: Yes |
| udwLineSpace | Specified line spacing in pixels |
| byRes | Reserved field. |

### Structure of window info

|  |
| --- |
| typedef struct tagNETDEVXWSceneWndInfo  {  UINT32 udwWndID;  UINT32 udwModifyCmd;  CHAR szWndName[NETDEV\_WND\_NAME\_LEN];  UINT32 udwPaneMod;  UINT32 udwLayer;  UINT32 udwTransparency;  [NETDEV\_XW\_AREA\_S](#_相对于电视墙的位置信息) stArea;  UINT32 udwZoomType;  UINT32 udwSplitIndex;  UINT32 udwSchemeResID;  UINT32 udwSeqStatus;  UINT32 udwPaneSize;  [NETDEV\_XW\_PANE\_INFO\_S](#_窗口分屏信息结构体) astPaneInfoList[NETDEV\_XW\_MAX\_PANE\_NUM];  UINT32 udwWndType;  UINT32 udwDChlID;  UINT32 udwStatus;  UINT32 udwScaleEnabled;  BYTE byRes[240];  }NETDEV\_XW\_SCENE\_WND\_INFO\_S, \*LPNETDEV\_XW\_SCENE\_WND\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwWndID | Window ID |
| udwModifyCmd | Modification command word. See [NETDEV\_XW\_WND\_CMD](#_场景窗口修改可选命令字枚举). |
| szWndName | Window name |
| udwPaneMod | Split window mode. Default : 1. See [NETDEV\_XW\_LAYOUT\_NUM\_E](#_屏幕输出制式枚举). |
| udwLayer | Layer. The layer with the greatest number is on the top. |
| udwTransparency | Degree of transparency. Range: 0-100. 0: Opaque (default) 100: Fully transparent |
| stArea | Location info |
| udwZoomType | Magnification type. See [NETDEV\_XW\_ZOOM\_TYPE\_E](#_窗口放大类型枚举). |
| udwSplitIndex | Split window ID. Valid when udwZoomType is NETDEV\_XW\_ZOOM\_TYPE\_SPLIT. |
| udwSchemeResID | Sequence resource ID |
| udwSeqStatus | Sequence status. See [NETDEV\_SEQ\_STATUS\_E](#_轮巡状态枚举_1). |
| udwPaneSize | Number of split windows |
| astPaneInfoList | All split window info |
| udwWndType | Window type. See [NETDEV\_TVWALL\_WND\_TPYE](#_窗口类型枚举). |
| udwDChlID | DC channel ID |
| udwStatus | 0: Not lock 1: Lock |
| udwScaleEnabled | Scale status. See [NETDEV\_SCALE\_ENABLED\_E](#_缩放状态枚举). Used only by LED sending and control device. Default: Full |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_CreateWnd](#_创建单个窗口配置信息), [NETDEV\_XW\_ModifyWnd](#_修改单个窗口配置信息), [NETDEV\_XW\_FindNextWnd](#_逐个获取查找到的窗口配置信息), [NETDEV\_XW\_GetWnd](#_获取单个窗口配置信息)

### Structure of split window info

|  |
| --- |
| typedef struct tagNETDEVXWPaneInfo  {  UINT32 udwPaneID;  INT32 dwChannelID;  UINT32 udwStreamIndex;  UINT32 udwTaskNo;  UINT32 udwTransProtocal;  UINT32 udwSessionType;  }NETDEV\_XW\_PANE\_INFO\_S, \*LPNETDEV\_XW\_PANE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPaneID | Split window ID |
| dwChannelID | Channel ID |
| udwStreamIndex | Stream index |
| udwTaskNo | Service ID |
| udwTransProtocal | Transport protocol. See [NETDEV\_PROTOCAL\_E](#_媒体传输协议枚举). |
| udwSessionType | Session type of split window. See [NETDEV\_SESSION\_TYPE\_E](#_分屏业务类型枚举) |

### Structure of window list info

|  |
| --- |
| typedef struct tagNETDEVXWBatchWindowInfoList  {  UINT32 udwOperation;  UINT32 udwNum;  [LPNETDEV\_XW\_SENCE\_BATCH\_WND\_S](#_批量开窗场景窗口信息结构体) pstSceneWinInfo;  BYTE byRes[256];  }NETDEV\_XW\_WND\_INFO\_LIST\_S,\*LPNETDEV\_XW\_WND\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwOperation | Batch window operation. 0: Roam windows in batches 1: Set split windows in batches |
| udwNum | Number of windows |
| pstSceneWinInfo | Window list |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_CreateBatchWnd](#_批量开窗), [NETDEV\_XW\_ModifyBatchWnd](#_批量修改窗口)

### Structure of window info for batch window opening

|  |
| --- |
| typedef struct tagNETDEVXWSenceBatchWnd  {  UINT32 udwReqSeq;  [NETDEV\_XW\_SCENE\_WND\_INFO\_S](#_窗口信息结构体) stSenceWndInfo;  BYTE byRes[32];  }NETDEV\_XW\_SENCE\_BATCH\_WND\_S, \*LPNETDEV\_XW\_SENCE\_BATCH\_WND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwReqSeq | Sequence number of request data |
| stSenceWndInfo | Window info |
| byRes | Reserved field. |

### Structure of info about batch operation results

|  |
| --- |
| typedef struct tagNETDEVXWSenceBatchResultList  {  UINT32 udwSize;  UINT32 udwLastChange;  [LPNETDEV\_XW\_BATCH\_RESULT\_WND\_S](#_批量开窗场景窗口返回信息结构体) pstResultInfo;  BYTE byRes[32];  }NETDEV\_XW\_BATCH\_RESULT\_LIST\_S, \*LPNETDEV\_XW\_BATCH\_RESULT\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSize | Number of windows |
| udwLastChange | Summary |
| pstResultInfo | Window info. Memory needs to be allocated dynamically according to udwSize. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_CreateBatchWnd](#_批量开窗), [NETDEV\_XW\_ModifyBatchWnd](#_批量修改窗口),

[NETDEV\_XW\_DeleteBatchWnd](#_批量关窗), [NETDEV\_AddACSPersonList](#_批量添加人员信息)

### Structure of returned result of opening windows in batches

|  |
| --- |
| typedef struct tagNETDEVXWSenceBatchResultWnd  {  UINT32 udwReqSeq;  UINT32 udwResuleCode;  UINT32 udwWinID;  }NETDEV\_XW\_BATCH\_RESULT\_WND\_S, \*LPNETDEV\_XW\_BATCH\_RESULT\_WND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwReqSeq | Sequence number of request data |
| udwResuleCode | Returned error code. |
| udwWinID | Window ID |

### Structure of screen zoom info

|  |
| --- |
| typedef struct tagNETDEVXWScreenZoomInfo  {  UINT32 uOperatedType;  UINT32 uOperatedMode;  UINT32 uSplitScreenIndex;  }NETDEV\_XW\_SCREEN\_ZOOM\_INFO\_S, \*LPNETDEV\_XW\_SCREEN\_ZOOM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| uOperatedType | Screen operation type. See [NETDEV\_XW\_SCREEN\_ZOOM\_TYPE\_E](#_屏幕放大类型枚举). |
| uOperatedMode | Screen operation mode. See [NETDEV\_XW\_SCREEN\_ZOOM\_MODE\_E](#_屏幕放大模式枚举). |
| uSplitScreenIndex | ID of the screen that needs to be magnified to the full screen. Valid when uOperatedType is 0 and uOperatedMode is 1. |

**See also:**

[NETDEV\_XW\_ZoomScreen](#_电视墙窗口放大/还原)

### Structure of decoding device ID info

|  |
| --- |
| typedef struct tagNETDEVXWDisplayerID  {  UINT32 udwTVWallID;  UINT32 udwWndID;  UINT32 udwPaneID;  BYTE byRes[16];  }NETDEV\_XW\_DISPLAYER\_ID\_S, \*LPNETDEV\_XW\_DISPLAYER\_ID\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTVWallID | Video wall ID |
| udwWndID | Window ID |
| udwPaneID | Split window ID |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_GetLiveBind](#_获取单个实况业务绑定信息), [NETDEV\_XW\_CreateLiveBind](#_绑定单个实况业务),

[NETDEV\_XW\_DeleteLiveBind](#_删除单个实况业务绑定), [NETDEV\_StartPassiveDecode](#_开始被动解码)

### Structure of video source info

|  |
| --- |
| typedef struct tagNETDEVXWVideoSource  {  [NETDEV\_XW\_DISPLAYER\_ID\_S](#_解码设备ID信息结构体) stDisplayerID;  UINT32 udwVideoInID;  CHAR szRealplayURL[NETDEV\_MAX\_URL\_LEN];  UINT32 udwStreamID;  [NETDEV\_AUTHENTICATION\_S](#_鉴权信息结构体) stAuthentication;  UINT32 udwTransProtocol;  UINT32 udwMulticast;  UINT32 udwType;  UINT32 udwDecodeType;  UINT32 udwManageType;  [LPNETDEV\_XW\_CONTENT\_INFO\_S](#_文字信息结构体) pstContentInfo;  BYTE byRes[56];  }NETDEV\_XW\_VIDEO\_SOURCE\_S, \*LPNETDEV\_XW\_VIDEO\_SOURCE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stDisplayerID | Device ID |
| udwVideoInID | Video source ID |
| szRealplayURL | Media stream URL. Configuration is not needed for local input. |
| udwStreamID | Media stream ID |
| stAuthentication | Authentication info |
| udwTransProtocol | Transport protocol. See [NETDEV\_LAPI\_TRANS\_PROTOCOL\_E](#_媒体传输协议枚举_2). Configuration is not needed for local input. |
| udwMulticast | Unicast or multicast. See [NETDEV\_STREAM\_TYPE\_E](#_流类型枚举). Optional for local input. |
| udwType | Live view type. See [NETDEV\_XW\_REALPLAY\_TYPE\_E](#_实况业务类型枚举). |
| udwDecodeType | Decoding type. See [NETDEV\_XW\_DECODE\_TYPE\_E](#_解码类型枚举). |
| udwManageType | Resource management type. See [NETDEV\_XW\_MANAGE\_TYPE\_E](#_资源管理类型枚举). |
| pstContentInfo | Text info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_GetLiveBind](#_获取单个实况业务绑定信息), [NETDEV\_XW\_CreateLiveBind](#_绑定单个实况业务),

[NETDEV\_XW\_FindNextLiveBindInfo](#_逐个获取实况业务绑定信息), [NETDEV\_StartPassiveDecode](#_开始被动解码)

### Structure of authentication info

|  |
| --- |
| typedef struct tagNETDEVAuthentication  {  BOOL bIsNeedAuthentication;  CHAR szUserName[NETDEV\_USER\_NAME\_ENCRYPT\_LEN];  CHAR szPassword[NETDEV\_PASSWORD\_ENCRYPT\_LEN];  [NETDEV\_AUTH\_MODEL\_E](#_加密类型枚举) enAuthModel;  BYTE byRes[64];  }NETDEV\_AUTHENTICATION\_S, \*LPNETDEV\_AUTHENTICATION\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bIsNeedAuthentication | Whether authentication is required |
| szUserName | Username for video source |
| szPassword | Password for video source |
| enAuthModel | Reserved field. Encryption type. See [NETDEV\_AUTH\_MODEL\_E](#_加密类型枚举). |
| byRes | Reserved field. |

### Structure of font info

|  |
| --- |
| typedef struct tagNETDEVXWContentInfo  {  CHAR szContent[NETDEV\_LEN\_260];  UINT32 udwFontType;  UINT32 udwFontSize;  UINT32 udwColor;  UINT32 udwPosition;  BYTE byRes[128];  }NETDEV\_XW\_CONTENT\_INFO\_S, \*LPNETDEV\_XW\_CONTENT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szContent | Text content |
| udwFontType | Font style. See [NETDEV\_XW\_LED\_FONT\_FORMAT\_E](#_虚拟LED字体枚举). |
| udwFontSize | Font size. 0: Adaptive. 48-1024: Font height in pixels |
| udwColor | Color in RGB: 0xABCDEF. AB means R (00~FF) CD means G (00~FF) EF means B (00~FF) Default: 0xFF0000 (red) |
| udwPosition | Font position. See [NETDEV\_XW\_CONTENT\_POSITION\_E](#_文字位置枚举). |
| byRes | Reserved field. |

### Structure of info about control parameters of sending passive decoding data

|  |
| --- |
| typedef struct tagNETDEVPassiveDecodeSend  {  INT32 dwTransType;  INT32 dwBufSize;  CHAR \*pszBuffer;  BYTE byRes[128];  }NETDEV\_PASSIVE\_SEND\_S, \*LPNETDEV\_PASSIVE\_SEND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwTransType | Transport type |
| dwBufSize | Data size |
| pszBuffer | Data content |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SendPassiveData](#_向被动解码通道发送数据)

### Structure of detailed stream session info

|  |
| --- |
| typedef struct tagNETDEVXWSteamInfo  {  UINT32 udwTaskNo;  UINT32 udwType;  [NETDEV\_XW\_STREAM\_INFO\_SRC\_S](#_源端媒体流信息结构体) stStreamInfoSrc;  [NETDEV\_XW\_STREAM\_INFO\_DEST\_S](#_目的端媒体流信息结构体) stStreamInfoDest;  [NETDEV\_XW\_STREAM\_INFO\_MSG\_S](#_媒体流具体信息结构体) stStreamInfoMsg;  }NETDEV\_XW\_STREAM\_INFO\_S, \*LPNETDEV\_XW\_STREAM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTaskNo | Service ID |
| udwType | Service type. See [NETDEV\_XW\_VIDEO\_TYPE\_E](#_流业务类型枚举). |
| stStreamInfoSrc | Source media stream info |
| stStreamInfoDest | Destination media stream info |
| stStreamInfoMsg | Detailed stream info |

**See also:**

[NETDEV\_XW\_FindNextStreamInfo](#_逐个获取业务流信息)

### Structure of basic info about sequence resource

|  |
| --- |
| typedef struct tagNETDEVXWSequenceResourceBaseInfo  {  UINT32 udwID;  CHAR szName[NETDEV\_LEN\_64];  CHAR szDescription[NETDEV\_LEN\_256];  BOOL bEnabled;  UINT32 udwOrderNO;  BYTE byRes[124];  }NETDEV\_XW\_SEQUENCE\_RES\_BASE\_INFO\_S, \*LPNETDEV\_XW\_SEQUENCE\_RES\_BASE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Unique ID of sequence resource |
| szName | Sequence resource name |
| szDescription | Sequence resource description |
| bEnabled | Enable or disable sequence resource |
| udwOrderNO | Sequence number of sequence resource. Required for query. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_GetSequenceResList](#_获取设备全部轮巡资源)

### Structure of detailed info about sequence resource

|  |
| --- |
| typedef struct tagNETDEVXWSequenceResDeTailInfo  {  [NETDEV\_XW\_SEQUENCE\_RES\_BASE\_INFO\_S](#_轮巡资源基本信息结构体) stSeqResBaseInfo;  UINT32 udwIntervalTime;  UINT32 udwVideoSrcNum;  [LPNETDEV\_XW\_SEQ\_RES\_VIDEO\_SRC\_INFO\_S](#_轮巡视频源信息结构体) pstVideoInList;  UINT32 udwManageType;  BYTE byRes[124];  }NETDEV\_XW\_SEQUENCE\_RES\_DETAIL\_INFO\_S, \*LPNETDEV\_XW\_SEQUENCE\_RES\_DETAIL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stSeqResBaseInfo | Basic info about sequence resource |
| udwIntervalTime | Sequence interval (unit: second) |
| udwVideoSrcNum | Number of video resources |
| pstVideoInList | Video source list. Memory needs to be allocated by the caller, and specifications are obtained through the capability interface. |
| udwManageType | Resource management type. See [NETDEV\_XW\_MANAGE\_TYPE\_E](#_资源管理类型枚举). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_AddSequenceResource](#_添加单个轮巡资源信息到设备), [NETDEV\_XW\_ModifySequenceResource](#_修改单个轮巡资源详细信息),

[NETDEV\_XW\_GetSequenceResource](#_获取单个轮巡资源详细信息)

### Structure of info about video sources of sequence resource

|  |
| --- |
| typedef struct tagNETDEVXWSeqVideoSrcInfo  {  UINT32 udwVideoSrcIndex;  [NETDEV\_XW\_VIDEO\_SOURCE\_BASE\_S](#_远端实况的视频源基本信息结构体) stVideoSrcBaseInfo;  UINT32 udwPresetIndex;  BYTE byRes[64];  }NETDEV\_XW\_SEQ\_RES\_VIDEO\_SRC\_INFO\_S, \*LPNETDEV\_XW\_SEQ\_RES\_VIDEO\_SRC\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwVideoSrcIndex | Sequence number of video source |
| stVideoSrcBaseInfo | Basic info about video source |
| udwPresetIndex | Preset ID. Configuration is required for sequence only. |
| byRes | Reserved field. |

### Structure of basic video source info for remote live video

|  |
| --- |
| typedef struct tagNETDEVXWVideoSourceBase  {  UINT32 udwVideoInID;  CHAR szRealplayURL[NETDEV\_MAX\_URL\_LEN];  UINT32 udwStreamID;  [NETDEV\_AUTHENTICATION\_S](#_鉴权信息结构体) stAuthentication;  UINT32 udwTransProtocol;  UINT32 udwMulticast;  UINT32 udwType;  UINT32 udwManageType;  BYTE byRes[28];  }NETDEV\_XW\_VIDEO\_SOURCE\_BASE\_S, \*LPNETDEV\_XW\_VIDEO\_SOURCE\_BASE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwVideoInID | Video source ID |
| szRealplayURL | Media stream URL. Configuration is not needed for local input. |
| udwStreamID | Media stream ID |
| stAuthentication | Authentication info |
| udwTransProtocol | Transport protocol. See [NETDEV\_LAPI\_TRANS\_PROTOCOL\_E](#_媒体传输协议枚举_2). Configuration is not needed for local input. |
| udwMulticast | Unicast or multicast. See [NETDEV\_STREAM\_TYPE\_E](#_流类型枚举). Optional for local input. |
| udwType | Live view type. See [NETDEV\_XW\_REALPLAY\_TYPE\_E](#_实况业务类型枚举). |
| udwManageType | Resource management type. See [NETDEV\_XW\_MANAGE\_TYPE\_E](#_资源管理类型枚举). |
| byRes | Reserved field. |

### Structure of basic scene info

|  |
| --- |
| typedef struct tagNETDEVXWSceneInfoBase  {  UINT32 udwSceneID;  CHAR szName[NETDEV\_SCENE\_NAME\_LEN];  UINT32 udwOrderNo;  UINT32 udwTVWallID;  UINT32 udwStatus;  BYTE byRes[20];  }NETDEV\_XW\_SCENE\_INFO\_BASE\_S, \*LPNETDEV\_XW\_SCENE\_INFO\_BASE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSceneID | Scene ID |
| szName | Scene name |
| udwOrderNo | Scene sequence number |
| udwTVWallID | Video wall ID |
| udwStatus | 0: Disable 1: Enable |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_CreateSceneInfo](#_创建单个场景信息)

### Structure of scene modification info

|  |
| --- |
| typedef struct tagNETDEVXWSceneModifyInfo  {  [NETDEV\_XW\_SCENE\_INFO\_BASE\_S](#_场景基本信息结构体) stSceneBaseInfo;  INT32 dwModifyType;  BYTE byRes[128];  }NETDEV\_XW\_SCENE\_MODIFY\_INFO\_S, \*LPNETDEV\_XW\_SCENE\_MODIFY\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stSceneBaseInfo | Scene info |
| dwModifyType | Modification type. See [NETDEV\_XW\_SCENE\_MODIFY\_TYPE\_E](#_场景修改类型枚举). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_ModifySceneInfo](#_修改场景信息)

### Structure of detailed scene info

|  |
| --- |
| typedef struct tagNETDEVXWSceneInfoDetail  {  CHAR szName[NETDEV\_SCENE\_NAME\_LEN];  UINT32 udwTVWallID;  CHAR szTVWallCode[NETDEV\_LEN\_64];  [NETDEV\_XW\_BASE\_MAP\_INFO\_S](#_所有底图信息结构体) stBaseMapInfo;  UINT32 udwVirtualLedNum;  [LPNETDEV\_XW\_VIRTUAL\_LED\_INFO\_S](#_虚拟LED信息结构体) pstVirtualLedInfo;  UINT32 udwWndNum;  [LPNETDEV\_XW\_SCENE\_WND\_INFO\_S](#_窗口信息结构体) pstWndInfo;  UINT32 udwLiveBindNum;  [LPNETDEV\_XW\_VIDEO\_SOURCE\_S](#_视频源信息结构体) pstVideoSource;  UINT32 udwSequenceBindNum;  [LPNETDEV\_XW\_SEQUENCE\_SOURCE\_LIST\_S](#_轮询资源列表信息结构体) pstSequenceSourceList;  UINT32 udwWindowsSequenceBindNum;  [LPNETDEV\_XW\_SEQUENCE\_RES\_BIND\_INFO\_S](#_轮巡资源绑定信息结构体) pstWindowsSequenceBindList;  UINT32 udwLinkedSceneID;  UINT32 udwLinkedSceneOrderNo;  UINT32 udwSceneSessionMode;  BYTE byRes[172];  }NETDEV\_XW\_SCENE\_INFO\_DETAIL\_S, \*LPNETDEV\_XW\_SCENE\_INFO\_DETAIL\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szName | Scene name |
| udwTVWallID | Video wall ID |
| szTVWallCode | Video wall code (optional). Assigned by device. String length range: [1,64] |
| stBaseMapInfo | Background image info |
| udwVirtualLedNum | Number of virtual LEDs |
| pstVirtualLedInfo | Virtual LED info |
| udwWndNum | Number of windows |
| pstWndInfo | Window info |
| udwLiveBindNum | Number of binding live view services |
| pstVideoSource | Info about binding live view services |
| udwSequenceBindNum | Number of binding sequence services |
| pstSequenceSourceList | Info about binding sequence services |
| udwWindowsSequenceBindNum | Number of binding sequence services |
| pstWindowsSequenceBindList | Detailed info about binding sequence services |
| udwLinkedSceneID | Linked scene ID |
| udwLinkedSceneOrderNo | Linked scene sequence number |
| udwSceneSessionMode | Session mode. See [NETDEV\_SCENE\_SESSION\_MODE\_E](#_场景业务模式枚举). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_GetCurrentSceneInfo](#_获取当前场景配置内容), [NETDEV\_XW\_GetSceneInfo](#_获取场景信息)

### Structure of all background image info

|  |
| --- |
| typedef struct tagNETDEVXWBaseMapInfo  {  UINT32 udwID;  CHAR szName[NETDEV\_BMAP\_NAME\_LEN];  UINT32 udwFormat;  [NETDEV\_XW\_RESOLUTION\_S](#_分辨率信息) stResolution;  UINT32 udwThumbnailData;  BYTE byRes[300];  }NETDEV\_XW\_BASE\_MAP\_INFO\_S, \*LPNETDEV\_XW\_BASE\_MAP\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Background image ID. Must be unique in the device. |
| szName | Background image name |
| udwFormat | Background image format. See [NETDEV\_XW\_BASE\_TYPE\_E](#_底图类型枚举). |
| stResolution | Resolution of background image |
| udwThumbnailData | Thumbnail image data |
| byRes | Reserved field. |

### Structure of sequence resource list

|  |
| --- |
| typedef struct tagNETDEVXWSequenceSourceList  {  UINT32 udwWndID;  UINT32 udwSourceSize;  [NETDEV\_XW\_SEQUENCE\_SOURCE\_S](#_轮询资源信息结构体) astSequenceSource[NETDEV\_SEQUENCE\_SRC\_MAX];  BYTE byRes[32];  }NETDEV\_XW\_SEQUENCE\_SOURCE\_LIST\_S, \*LPNETDEV\_XW\_SEQUENCE\_SOURCE\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwWndID | Window ID |
| udwSourceSize | Number of video sources in camera service |
| astSequenceSource | Information about video sources in camera service |
| byRes | Reserved field. |

### Structure of sequence resource info

|  |
| --- |
| typedef struct tagNETDEVXWSequenceSource  {  UINT32 udwInterval;  UINT32 udwPreset;  [NETDEV\_XW\_VIDEO\_SOURCE\_BASE\_S](#_远端实况的视频源基本信息结构体) stVideoSourceBase;  BYTE byRes[32];  }NETDEV\_XW\_SEQUENCE\_SOURCE\_S, \*LPNETDEV\_XW\_SEQUENCE\_SOURCE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwInterval | Sequence interval (unit: second) |
| udwPreset | Preset ID |
| stVideoSourceBase | Information about video sources in camera service |
| byRes | Reserved field. |

### Structure of brief scene info

|  |
| --- |
| typedef struct tagNETDEVXWSceneInfoShort  {  [NETDEV\_XW\_SCENE\_INFO\_BASE\_S](#_场景基本信息结构体) stSceneBase;  UINT32 udwIsCurrent;  BYTE byRes[64];  }NETDEV\_XW\_SCENE\_INFO\_BASIC\_S, \*LPNETDEV\_XW\_SCENE\_INFO\_BASIC\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stSceneBase | Basic scene info |
| udwIsCurrent | Whether is the current scene. 0: No 1: Yes |
| byRes | Reserved field. |

**See also:**

[[NETDEV\_XW\_FindNextSceneCfg](#_逐个获取查找到的场景配置信息)](#_逐个获取查找到的场景配置信息)

### Structure of sequence resource binding info

|  |
| --- |
| typedef struct tagNETDEVXWSequenceResourceBindBaseInfo  {  UINT32 udwID;  UINT32 udwSequenceResID;  UINT32 udwIntervalTime;  UINT32 udwStatus;  UINT32 udwWndNum;  UINT32 audwWndIDList[NETDEV\_SEQ\_RES\_WIN\_MAX];  BYTE byRes[256];  }NETDEV\_XW\_SEQUENCE\_RES\_BIND\_INFO\_S, \*LPNETDEV\_XW\_SEQUENCE\_RES\_BIND\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Sequence resource binding ID. Optional when adding sequence resource. |
| udwSequenceResID | Sequence resource ID |
| udwIntervalTime | Sequence time interval (unit: second) |
| udwStatus | Sequence status. See [NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_E](#_轮巡状态枚举). |
| udwWndNum | Number of windows |
| audwWndIDList | Window ID list. Memory needs to be allocated by user, and specifications are obtained through the capability interface. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_BindSequenceResToScene](#_绑定轮巡资源到当前场景布局), [NETDEV\_XW\_ModifySequenceResToScene](#_修改当前场景轮巡布局)

### Structure of info about sequence play control

|  |
| --- |
| typedef struct tagNETDEVXWSequencePlayControlInfo  {  UINT32 udwSequenceResBindID;  UINT32 udwOperation;  BYTE byRes[128];  }NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_INFO\_S, \*LPNETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSequenceResBindID | Sequence resource binding ID |
| udwOperation | Operation command type. See [NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_E](#_轮巡播放控制枚举) |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_SequencePlayControl](#_轮巡播放控制)

### Structure of basic info about scene sequence plan

|  |
| --- |
| typedef struct tagNETDEVXWScenesPlanInfo  {  UINT32 udwID;  CHAR szName[NETDEV\_LEN\_64];  UINT32 udwOrderNo;  CHAR szDescription[NETDEV\_LEN\_256];  UINT32 udwStatus;  UINT32 udwMode;  [NETDEV\_XW\_SCENES\_PLAN\_TIMER\_INFO\_S](#_场景计划定时切换模式信息结构体) stTimerInfo;  [NETDEV\_XW\_SCENES\_PLAN\_TIME\_TABLE\_INFO\_S](#_场景计划时间表模式信息结构体) stTimeTableInfo;  BYTE byRes[124];  }NETDEV\_XW\_SCENES\_PLAN\_INFO\_S, \*LPNETDEV\_XW\_SCENES\_PLAN\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Plan ID |
| szName | Plan name |
| udwOrderNo | Sequence number of plan |
| szDescription | Plan description |
| udwStatus | Plan status. See [NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_E](#_轮巡状态枚举). |
| udwMode | Plan mode. See [NETDEV\_XW\_SCENES\_PLAN\_MODE\_E](#_场景轮巡计划模式枚举). |
| stTimerInfo | Info about timed switching |
| stTimeTableInfo | Time table mode info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_AddSencesSequencePlan](#_添加单个场景轮巡计划), [NETDEV\_XW\_ModifySencesSequencePlan](#_修改单个场景轮巡计划),

[NETDEV\_XW\_GetSencesSequencePlan](#_获取单个场景轮巡计划), [NETDEV\_XW\_FindNextSencesSequencePlan](#_逐个获取场景轮巡计划)

### Structure of info about scene switching timer

|  |
| --- |
| typedef struct tagNETDEVXWScenesPlanTimerInfo  {  UINT32 udwIntervalTime;  UINT32 udwSceneNum;  UINT32\* pudwSceneIDList;  BYTE byRes[64];  }NETDEV\_XW\_SCENES\_PLAN\_TIMER\_INFO\_S, \*LPNETDEV\_XW\_SCENES\_PLAN\_TIMER\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwIntervalTime | Time interval (unit: second) |
| udwSceneNum | Number of scenes |
| pudwSceneIDList | Scene ID list. Memory needs to be allocated by user. |
| byRes | Reserved field. |

### Structure of info about scene switching time table

|  |
| --- |
| typedef struct tagNETDEVXWScenesPlanTimeTableInfo  {  UINT32 udwTimePeriodNum;  [LPNETDEV\_XW\_TIME\_TABLE\_INFO\_S](#_24小时时间表配置信息结构体) pstTimeTableList;  BYTE byRes[64];  }NETDEV\_XW\_SCENES\_PLAN\_TIME\_TABLE\_INFO\_S, \*LPNETDEV\_XW\_SCENES\_PLAN\_TIME\_TABLE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTimePeriodNum | Number of time periods |
| pstTimeTableList | Time table list. Memory needs to be allocated by user according to udwTimePeriodNum. |
| byRes | Reserved field. |

### Structure of info about 24-hour time table

|  |
| --- |
| typedef struct tagNETDEVXWTimeTableInfo  {  UINT32 udwSceneID;  CHAR szStartTime[NETDEV\_LEN\_16];  BYTE byRes[64];  }NETDEV\_XW\_TIME\_TABLE\_INFO\_S, \*LPNETDEV\_XW\_TIME\_TABLE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSceneID | Scene ID |
| szStartTime | Start time (hh:mm:ss) |
| byRes | Reserved field. |

### Structure of info about scene sequence plan playing control

|  |
| --- |
| typedef struct tagNETDEVXWScenesSeqPlanPlayCtrlInfo  {  UINT32 udwScenesSeqPlanID;  UINT32 udwOperation;  BYTE byRes[128];  }NETDEV\_XW\_SCENES\_PLAN\_PLAY\_CTRL\_INFO\_S, \*LPNETDEV\_XW\_SCENES\_PLAN\_PLAY\_CTRL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwScenesSeqPlanID | Scene sequence plan ID |
| udwOperation | Operation command type. See [NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_E](#_轮巡播放控制枚举) |
| byRes | Reserved field. |

**See also:**

[NETDEV\_XW\_ScenesSeqPlanPlayControl](#_场景轮巡计划播放控制)

### Structure of stream source info

|  |
| --- |
| typedef struct tagNETDEVXWSteamInfoSrc  {  UINT32 udwVideoInChlID;  UINT32 udwStreamID;  CHAR szIPAddr[NETDEV\_IPV4\_LEN\_MAX];  UINT32 udwPort;  BYTE byRes[64];  }NETDEV\_XW\_STREAM\_INFO\_SRC\_S, \*LPNETDEV\_XW\_STREAM\_INFO\_SRC\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwVideoInChlID | Video source ID |
| udwStreamID | Media stream ID |
| szIPAddr | Video source IP |
| udwPort | Port number |
| byRes | Reserved field. |

### Structure of stream destination info

|  |
| --- |
| typedef struct tagNETDEVXWSteamInfoDest  {  UINT32 udwWndID;  UINT32 udwPaneID;  CHAR szIPAddr[NETDEV\_IPV4\_LEN\_MAX];  UINT32 udwPort;  BYTE byRes[64];  }NETDEV\_XW\_STREAM\_INFO\_DEST\_S, \*LPNETDEV\_XW\_STREAM\_INFO\_DEST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwWndID | Window ID |
| udwPaneID | Split window ID |
| szIPAddr | Destination IP |
| udwPort | Port number |
| byRes | Reserved field. |

### Structure of detailed stream info

|  |
| --- |
| typedef struct tagNETDEVXWStreamInfoMsg  {  UINT32 udwTransProtocol;  UINT32 udwIsMulticast;  [NETDEV\_XW\_RESOLUTION\_S](#_分辨率信息) stResolution;  UINT32 udwFrame;  UINT32 udwEncodeFormat;  UINT32 udwIsInvalidAfterReboot;  BYTE byRes[64];  }NETDEV\_XW\_STREAM\_INFO\_MSG\_S, \*LPNETDEV\_XW\_STREAM\_INFO\_MSG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTransProtocol | Transport protocol. See [NETDEV\_LAPI\_TRANS\_PROTOCOL\_E](#_媒体传输协议枚举_2). |
| udwIsMulticast | Unicast or multicast. See [NETDEV\_STREAM\_TYPE\_E](#_流类型枚举). |
| stResolution | Resolution |
| udwFrame | Frame rate |
| udwEncodeFormat | Encoding format. See [NETDEV\_XW\_VIDEOENCODING\_E](#_编码格式枚举). |
| udwIsInvalidAfterReboot | Whether device needs to be restored. 0: not restore Xware device 1: restore Xware device |
| byRes | Reserved field. |

### Structure of manual alarm linkage

|  |
| --- |
| typedef struct tagNETDEVManualLink  {  UINT32 udwType;  [NETDEV\_POINT\_S](#_点坐标结构体) stPoint;  [NETDEV\_RECTANGLE\_POINT\_S](#_矩形坐标结构体) stRectangle;  BYTE byRes[128];  }NETDEV\_MANUAL\_LINK\_S,\*LPNETDEV\_MANUAL\_LINK\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | Linkage type. See [NETDEV\_LINKAGE\_TYPE\_E](#_手动联动类型枚举). |
| stPoint | Point coordinates. Valid when Type is 0 or 2. |
| stRectangle | Rectangle coordinates. Valid when Type is 1. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_ManualLink](#_手动联动操作)

### Structure of rectangle coordinates

|  |
| --- |
| typedef struct tagNETDEVRectanglePoint  {  [NETDEV\_POINT\_S](#_点坐标结构体) stTopLeft;  [NETDEV\_POINT\_S](#_点坐标结构体) stBottomRight;  BYTE byRes[256];  }NETDEV\_RECTANGLE\_POINT\_S, \*LPNETDEV\_RECTANGLE\_POINT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stTopLeft | Coordinates of top left corner. |
| stBottomRight | Coordinates of bottom right corner. |
| byRes | Reserved field. |

### Structure of plate recognition data info

|  |
| --- |
| typedef struct tagNETDEVTMSCarPlateInfo  {  UINT32 udwPicNum;  [NETDEV\_TMS\_CAR\_PLATE\_XML\_INFO\_S](#_车牌识别报文XML解析信息结构体) stTmsXmlInfo;  [NETDEV\_TMS\_CAR\_PLATE\_PIC\_INFO\_S](#_车牌识别报文图片解析信息结构体) stTmsPicInfo[NETDEV\_TMS\_PIC\_COMMON\_NUM];  }NETDEV\_TMS\_CAR\_PLATE\_INFO\_S,\*LPNETDEV\_TMS\_CAR\_PLATE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPicNum | Number of images |
| stTmsXmlInfo | XML info |
| stTmsPicInfo | Image info |

**See also:**

[NETDEV\_CarPlateCallBack\_PF](#_注册车牌识别上报回调函数)

### Structure of plate recognition xml info

|  |
| --- |
| typedef struct tagNETDEVTMSCarPlateXmlInfo  {  CHAR szCamID[NETDEV\_TMS\_CAR\_PLATE\_CAMID\_LEN];  CHAR szRecordID[NETDEV\_TMS\_CAR\_PLATE\_RECORDID\_LEN];  CHAR szTollgateID[NETDEV\_TMS\_CAR\_PLATE\_TOLLGATE\_LEN];  CHAR szPassTime[NETDEV\_TMS\_CAR\_PLATE\_PASSTIME\_LEN];  CHAR szLaneID[NETDEV\_TMS\_CAR\_PLATE\_LANEID\_LEN];  CHAR szCarPlate[NETDEV\_TMS\_CAR\_PLATE\_CARPLATE\_LEN];  CHAR szIPAddr[NETDEV\_IPV4\_LEN\_MAX];  INT32 dwCarPlateColor;  BYTE bRes[492];  }NETDEV\_TMS\_CAR\_PLATE\_XML\_INFO\_S, \*LPNETDEV\_TMS\_CAR\_PLATE\_XML\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szCamID | Checkpoint camera ID |
| szRecordID | Record ID |
| szTollgateID | Checkpoint ID |
| szPassTime | Pass-through time |
| szLaneID | Lane ID |
| szCarPlate | Plate No. |
| szIPAddr | Device IP address |
| dwCarPlateColor | Plate color. See [NETDEV\_TMS\_CAR\_PLATE\_COLOR\_E](#_号牌颜色类型枚举). |
| bRes | Reserved field. |

### Structure of plate recognition image info

|  |
| --- |
| typedef struct tagNETDEVTMSCarPlatePicInfo  {  UINT32 udwPicSize;  CHAR \*pcPicData;  }NETDEV\_TMS\_CAR\_PLATE\_PIC\_INFO\_S,\*LPNETDEV\_TMS\_CAR\_PLATE\_PIC\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPicSize | Image size |
| pcPicData | Image data |

### Structure of face recognition record

|  |
| --- |
| typedef struct tagstNETDEVFaceRecordSnapshotInfo  {  UINT32 udwRecordID;  UINT32 udwRecordType;  UINT32 udwPassTime;  UINT32 udwEventType;  UINT32 udwChannelID;  UINT32 udwMonitorRuleID;  CHAR szChannelName[NETDEV\_LEN\_260];  NETDEV\_FACE\_ALARM\_CMP\_INFO\_S stCompareInfo;  BYTE byRes[124];  }NETDEV\_FACE\_RECORD\_SNAPSHOT\_INFO\_S, \*LPNETDEV\_FACE\_RECORD\_SNAPSHOT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwRecordID | Face recognition record ID |
| udwRecordType | Face recognition record type. See [NETDEV\_FACE\_PASS\_RECORD\_TYPE\_E](#_人脸通行记录类型枚举). |
| udwPassTime | Person pass-through time (UTC format) |
| udwEventType | Event type. Described by BIT. BIT=1 means this type; BIT=0 means not this type. BIT0: face image capture BIT1: face match alarm BIT2: face not match alarm |
| udwChannelID | Channel ID |
| udwMonitorRuleID | Corresponding monitoring task ID of the alarm |
| szChannelName | Capture channel name |
| stCompareInfo | Comparison info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_FaceAlarmMessCallBack\_PF](#_注册人脸识别报警消息回调函数), [[NETDEV\_FindNextFaceRecordDetail](#_逐个获取人脸识别记录)](#_逐个获取人脸识别记录)

### Structure of info about face snapshot comparison

|  |
| --- |
| typedef struct tagstNETDEVFaceAlarmCmpInfo  {  UINT32 udwSimilarity;  [NETDEV\_FACE\_MEMBER\_INFO\_S](#_人脸库成员信息结构体) stMemberInfo;  [NETDEV\_FACE\_ALARM\_SNAP\_IMAGE\_S](#_抓拍图片信息结构体_1) stSnapshotImage;  [LPNETDEV\_PERSON\_INFO\_S](#_人员信息结构体) pstPersonInfo;  [LPNETDEV\_FACE\_ATTR\_S](#_人脸属性信息) pstFaceAttr;  [LPNETDEV\_PERSON\_ATTR\_S](#_人员属性) pstPersonAttr;  BYTE byRes[116];  }NETDEV\_FACE\_ALARM\_CMP\_INFO\_S, \*LPNETDEV\_FACE\_ALARM\_CMP\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSimilarity | Matching degree (similarity) |
| stMemberInfo | Face library member info |
| stSnapshotImage | Snapshot image |
| pstPersonInfo | Face library member info (applicable to NVR). Required when getting match/mismatch records. Memory needs to be allocated dynamically. |
| pstFaceAttr | Face attribute info. Memory needs to be allocated dynamically. |
| pstPersonAttr | Person attribute info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Structure of face library member info

|  |
| --- |
| typedef struct tagNETDEVFaceMemberInfo  {  UINT32 udwReqSeq;  UINT32 udwMemberID;  CHAR szMemberName[NETDEV\_FACE\_MEMBER\_NAME\_LEN];  UINT32 udwMemberGender;  CHAR szMemberBirthday[NETDEV\_FACE\_MEMBER\_BIRTHDAY\_LEN];  [NETDEV\_FACE\_MEMBER\_REGION\_INFO\_S](#_人脸库成员地区信息结构体) stMemberRegionInfo;  [NETDEV\_FACE\_MEMBER\_ID\_INFO\_S](#_成员证件信息结构体_1) stMemberIDInfo;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stMemberImageInfo;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stMemberSemiInfo;  UINT32 udwCustomNum;  [NETDEV\_CUSTOM\_VALUE\_S](#_自定义属性信息结构体) stCustomValue[NETDEV\_FACE\_MEMBER\_CUSTOM\_NUM];  BOOL bIsMonitored;  UINT32 udwDBNum;  UINT32 audwDBIDList[NETDEV\_LEN\_16];  BYTE byRes[128];  }NETDEV\_FACE\_MEMBER\_INFO\_S,\*LPNETDEV\_FACE\_MEMBER\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwReqSeq | Sequence number of request data |
| udwMemberID | Face library member ID. Optional in read-only post messages. |
| szMemberName | Member name |
| udwMemberGender | Member gender. See [NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举) |
| szMemberBirthday | Birth date |
| stMemberRegionInfo | Region info |
| stMemberIDInfo | Member ID info |
| stMemberImageInfo | Face image info |
| stMemberSemiInfo | Semi-structured face info |
| udwCustomNum | Number of custom attributes |
| stCustomValue | Custom attribute list |
| bIsMonitored | Whether monitoring is configured 0: no 1: yes |
| udwDBNum | Number of face libraries containing the member |
| audwDBIDList | Face library ID list |
| byRes | Reserved field. |

### Structure of snapshot image info

|  |
| --- |
| typedef struct tagstNETDEVFaceAlarmSnapImage  {  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stBigImage;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stSmallImage;  [NETDEV\_FACE\_ALARM\_IMAGE\_AREA\_S](#_区域坐标结构体) stArea;  BYTE byRes[128];  }NETDEV\_FACE\_ALARM\_SNAP\_IMAGE\_S, \*LPNETDEV\_FACE\_ALARM\_SNAP\_IMAGE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stBigImage | Complete image |
| stSmallImage | Small image |
| stArea | Coordinates of area |
| byRes | Reserved field. |

### Structure of region info of face library members

|  |
| --- |
| typedef struct tagNETDEVFaceMemberRegionInfo  {  CHAR szNation[NETDEV\_FACE\_MEMBER\_REGION\_LEN];  CHAR szProvince[NETDEV\_FACE\_MEMBER\_REGION\_LEN];  CHAR szCity[NETDEV\_FACE\_MEMBER\_REGION\_LEN];  BYTE byRes[256];  }NETDEV\_FACE\_MEMBER\_REGION\_INFO\_S,\*LPNETDEV\_FACE\_MEMBER\_REGION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szNation | Nationality |
| szProvince | Province |
| szCity | City |
| byRes | Reserved field. |

### Structure of member’s ID info

|  |
| --- |
| typedef struct tagNETDEVFaceMemberIDInfo  {  UINT32 udwType;  CHAR szNumber[NETDEV\_FACE\_IDNUMBER\_LEN];  BYTE byRes[128];  }NETDEV\_FACE\_MEMBER\_ID\_INFO\_S,\*LPNETDEV\_FACE\_MEMBER\_ID\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | ID type. See [NETDEV\_FACE\_MEMBER\_ID\_TYPE\_E](#_人脸成员证件类型枚举). |
| szNumber | ID No. |
| byRes | Reserved field. |

### Structure of area coordinates

|  |
| --- |
| typedef struct tagstNETDEVFaceAlarmImageArea  {  UINT32 udwLeft;  UINT32 udwTop;  UINT32 udwRight;  UINT32 udwButtom;  BYTE byRes[128];  }NETDEV\_FACE\_ALARM\_IMAGE\_AREA\_S, \*LPNETDEV\_FACE\_ALARM\_IMAGE\_AREA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwLeft | Left coordinates |
| udwTop | Top coordinates |
| udwRight | Right coordinates |
| udwButtom | Bottom coordinates |

### Structure of face snapshot info

|  |
| --- |
| typedef struct tagNETDEVTMSFaceSnapshotPicInfo  {  UINT32 udwFaceId;  CHAR \*pcPicBuff;  UINT32 udwPicBuffLen;  [NETDEV\_TMS\_PERSION\_IMAGE\_TYPE\_E](#_图片类型枚举枚举) enImgType;  [NETDEV\_TMS\_PERSION\_IMAGE\_FORMAT\_E](#_图片格式枚举枚举) enImgFormat;  [NETDEV\_FACE\_POSITION\_INFO\_S](#_人脸位置信息结构体) stFacePos;  UINT32 udwImageWidth;  UINT32 udwImageHeight;  CHAR szCamerID[NETDEV\_TMS\_CAMER\_ID\_LEN];  CHAR szRecordID[NETDEV\_TMS\_FACE\_RECORD\_ID\_LEN];  CHAR szTollgateID[NETDEV\_TMS\_FACE\_TOLLGATE\_ID\_LEN];  CHAR szPassTime[NETDEV\_TMS\_PASSTIME\_LEN];  UINT32 udwFaceNum;  CHAR szIPAddr[NETDEV\_IPV4\_LEN\_MAX];  BYTE bRes[76];  }NETDEV\_TMS\_FACE\_SNAPSHOT\_PIC\_INFO\_S, \*LPNETDEV\_TMS\_FACE\_SNAPSHOT\_PIC\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwFaceId | Face ID |
| pcPicBuff | Image buffer |
| udwPicBuffLen | Image buffer length |
| enImgType | Image type. See [NETDEV\_TMS\_PERSION\_IMAGE\_TYPE\_E](#_图片类型枚举枚举). |
| enImgFormat | Image format. See [NETDEV\_TMS\_PERSION\_IMAGE\_FORMAT\_E](#_图片格式枚举). |
| stFacePos | Coordinates of face and image are normalized: 0-10000. Top left and bottom right of the rectangular: "138,315,282,684". |
| udwImageWidth | Image width |
| udwImageHeight | Image height |
| szCamerID | Camera ID |
| szRecordID | Record ID |
| szTollgateID | Checkpoint ID |
| szPassTime | Pass-through time (YYYYMMDDHHMMSSMMM, 24H format). The first MM represents month, the second MM represents minute, and the third MMM represents millisecond |
| udwFaceNum | Number of faces. Valid in panoramic images. |
| szIPAddr | Device IP |
| bRes | Reserved field. |

**See also:**

[NETDEV\_FaceSnapshotCallBack\_PF](#_注册人脸抓拍上报回调函数)

### Structure of face position info

|  |
| --- |
| typedef struct tagNETDEVFacePositionInfo  {  INT32 dwTopLeftX;  INT32 dwTopLeftY;  INT32 dwBottomRightX;  INT32 dwBottomRightY;  BYTE bRes[32];  }NETDEV\_FACE\_POSITION\_INFO\_S, \*LPNETDEV\_FACE\_POSITION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwTopLeftX | x-coordinate of top left corner [0, 10000] |
| dwTopLeftY | y-coordinate of top left corner [0, 10000] |
| dwBottomRightX | x-coordinate of bottom right corner [0, 10000] |
| dwBottomRightY | y-coordinate of bottom right corner [0, 10000] |
| bRes | Reserved field. |

### Structure of heatmap data info

|  |
| --- |
| typedef struct tagNETDEVTMSHeatMapInfo  {  UINT32 udwPicNum;  [NETDEV\_TMS\_HEAT\_MAP\_XML\_INFO\_S](#_热度图报文XML解析信息结构体) stTmsXmlInfo;  [NETDEV\_TMS\_HEAT\_MAP\_PIC\_INFO\_S](#_热度图报文区域解析信息结构体) stTmsPicInfo[NETDEV\_TMS\_PIC\_COMMON\_NUM];  }NETDEV\_TMS\_HEAT\_MAP\_INFO\_S,\*LPNETDEV\_TMS\_HEAT\_MAP\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPicNum | Number of areas |
| stTmsXmlInfo | XML info |
| stTmsPicInfo | Area info |

**See also:**

[NETDEV\_HeatMapCallBack\_PF](#_注册热度图上报回调函数)

### Structure of heatmap xml info

|  |
| --- |
| typedef struct tagNETDEVTMSHeatMapXmlInfo  {  CHAR szDevID[NETDEV\_TMS\_HEAT\_MAP\_DEVID\_LEN];  CHAR szRecordID[NETDEV\_TMS\_HEAT\_MAP\_RECORD\_ID\_LEN];  CHAR szCollectTime[NETDEV\_TMS\_HEAT\_MAP\_COllECT\_TIME\_LEN];  INT32 dwColumns;  INT32 dwRows;  INT32 dwIntervalTime;  CHAR szIPAddr[NETDEV\_IPV4\_LEN\_MAX];  BYTE bRes[240];  }NETDEV\_TMS\_HEAT\_MAP\_XML\_INFO\_S, \*LPNETDEV\_TMS\_HEAT\_MAP\_XML\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szDevID | Device ID |
| szRecordID | Record ID |
| szCollectTime | Collection time |
| dwColumns | Number of detection columns |
| dwRows | Number of detection rows |
| dwIntervalTime | Detection interval |
| szIPAddr | Device IP |
| bRes | Reserved field. |

### Structure of heatmap area info

|  |
| --- |
| typedef struct tagNETDEVTMSHeatMapPicInfo  {  UINT32 udwPicSize;  CHAR \*pcPicData;  }NETDEV\_TMS\_HEAT\_MAP\_PIC\_INFO\_S,\*LPNETDEV\_TMS\_HEAT\_MAP\_PIC\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPicSize | Area size |
| pcPicData | Area data |

### Person alarm info

|  |
| --- |
| typedef struct tagNETDEVPersonEventInfo  {  UINT32 udwID;  UINT32 udwTimestamp;  UINT32 udwNotificationType;  UINT32 udwFaceInfoNum;  [NETDEV\_FACE\_PASS\_RECORD\_INFO\_S](#_人脸通过记录信息) stCtrlFaceInfo[NETDEV\_LEN\_2];  UINT32 udwFinishFaceNum;  UINT32 audwFinishFaceList[NETDEV\_LEN\_40];  CHAR szReference[NETDEV\_LEN\_480];  BYTE byRes[92];  }NETDEV\_PERSON\_EVENT\_INFO\_S, \*LPNETDEV\_PERSON\_EVENT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Notification record ID |
| udwTimestamp | Notification reporting time (unit: second, UTC format). |
| udwNotificationType | Notification type. 0: Real-time notification 1: Historical notification |
| udwFaceInfoNum | Number of face info. Range: [0, 1] |
| stCtrlFaceInfo | Face list. Optional if no face is collected. |
| udwFinishFaceNum | Number of finish faces. Range: [0, 40] |
| audwFinishFaceList | Finish face list |
| szReference | Descriptive info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_PersonAlarmMessCallBack\_PF](#_注册速人脸报警消息回调函数)

### Face pass-through records info

|  |
| --- |
| typedef struct tagNETDEVFacePassRecordInfo  {  UINT32 udwRecordID;  UINT32 udwType;  INT64 tPassingTime;  UINT32 udwChannelID;  CHAR szChlName[NETDEV\_LEN\_128];  [NETDEV\_PERSON\_COMPARE\_INFO\_S](#_人脸比对信息) stCompareInfo;  BYTE byRes[256];  }NETDEV\_FACE\_PASS\_RECORD\_INFO\_S, \*LPNETDEV\_FACE\_PASS\_RECORD\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwRecordID | Face pass-thru record ID |
| udwType | Face pass-thru type. See [NETDEV\_FACE\_PASS\_RECORD\_TYPE\_E](#_人脸通行记录类型枚举) |
| tPassingTime | Person pass-through time (unit: second, UTC format). |
| udwChannelID | Channel ID |
| szChlName | Capture channel name. Range: [1, 63]. |
| stCompareInfo | Comparison info |
| byRes | Reserved field. |

### Face comparison info

|  |
| --- |
| typedef struct tagNETDEVPersonCompareInfo  {  UINT32 udwSimilarity;  [NETDEV\_PERSON\_INFO\_S](#_人员信息结构体) stPersonInfo;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stPanoImage;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stFaceImage;  [NETDEV\_FACE\_POSITION\_INFO\_S](#_人脸位置信息结构体) stFaceArea;  UINT32 udwCapSrc;  UINT32 udwFeatureNum;  [LPNETDEV\_FEATURE\_INFO\_S](#_半结构化特征信息) pstFeatureInfo;  [NETDEV\_FACE\_ATTR\_S](#_人脸属性信息) stFaceAttr;  [NETDEV\_PERSON\_ATTR\_S](#_人员属性) stPersonAttr;  BYTE byRes[248];  }NETDEV\_PERSON\_COMPARE\_INFO\_S, \*LPNETDEV\_PERSON\_COMPARE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSimilarity | Matching degree (similarity) |
| stPersonInfo | Face library member info |
| stPanoImage | Complete image |
| stFaceImage | Face image |
| stFaceArea | Face area coordinates |
| udwCapSrc | Source of collection |
| udwFeatureNum | Number of semi-structured attributes. Optional if semi-structured attributes are unavailable. Required for PTS. |
| pstFeatureInfo | Semi-structured attribute list. Optional if semi-structured attributes are unavailable. Required for PTS. |
| stFaceAttr | Face attribute |
| stPersonAttr | Attribute of the related person |
| byRes | Reserved field. |

### Semi-structured attribute info

|  |
| --- |
| typedef struct tagNETDEVFeatureInfo  {  CHAR szFeatureVersion[NETDEV\_LEN\_32];  CHAR szFeature[NETDEV\_FACE\_FEATURE\_SIZE];  BYTE byRes[128];  }NETDEV\_FEATURE\_INFO\_S,\*LPNETDEV\_FEATURE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szFeatureVersion | Version number of the algorithm extracting semi-structured attributes. Range: [0, 20] |
| szFeature | Extracted face attribute info. 512 bytes before encryption. |
| byRes | Reserved field |

### Face attribute info

|  |
| --- |
| typedef struct tagNETDEVFaceAttr  {  UINT32 udwGender;  UINT32 udwAgeRange;  UINT32 udwGlassFlag;  UINT32 udwGlassesStyle;  UINT32 udwMask;  FLOAT fTemperature;  UINT32 udwEmotion;  UINT32 udwSmile;  UINT32 udwAttractive;  UINT32 udwSkinColor;  UINT32 udwBeard;  BYTE byRes[112];  }NETDEV\_FACE\_ATTR\_S,\*LPNETDEV\_FACE\_ATTR\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwGender | Gender. See [NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举). |
| udwAgeRange | Age. See [NETDEV\_AGE\_RANGE\_E](#_年龄段枚举). |
| udwGlassFlag | Whether wearing glasses. See [NETDEV\_GLASS\_FLAG\_E](#_是否戴眼镜标志枚举). |
| udwGlassesStyle | Glasses style. See [NETDEV\_GLASSES\_STYLE\_E](#_眼镜款式枚举). |
| udwMask | Mask. See [[NETDEV\_MASK\_FLAG\_E](#_是否戴口罩枚举)](#_是否戴口罩枚举_1). |
| fTemperature | Body temperature (unit: °C, 2 decimal places) |
| udwEmotion | Emotion. Optional when not detected. See [NETDEV\_EMOTION\_FLAG\_E](#_情绪情况). |
| udwSmile | Smile. Optional when not detected. See [NETDEV\_SMILE\_FLAG\_E](#_微笑标志). |
| udwAttractive | Facial attractiveness. Optional when not detected. Range: [0~100] |
| udwSkinColor | Skin color. Optional when not detected. See [NETDEV\_SKINCOLOR\_TYPE\_E](#_肤色枚举). |
| udwBeard | Beard. Optional when not detected. See [NETDEV\_BEARD\_FLAG\_E](#_胡子标志). |
| byRes | Reserved field. |

### Person attributes

|  |
| --- |
| typedef struct tagNETDEVPersonAttr  {  UINT32 udwGender;  UINT32 udwAgeRange;  UINT32 udwSleevesLength;  UINT32 udwCoatColor;  UINT32 udwTrousersLength;  UINT32 udwTrousersColor;  UINT32 udwBodyToward;  UINT32 udwShoesTubeLength;  UINT32 udwHairLength;  UINT32 udwBagFlag;  FLOAT fTemperature;  UINT32 udwMask;  UINT32 udwCoatTexture;  UINT32 udwMovingDirection;  BYTE byRes[116];  }NETDEV\_PERSON\_ATTR\_S,\*LPNETDEV\_PERSON\_ATTR\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwGender | Gender. See [NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举). |
| udwAgeRange | Age. See [NETDEV\_AGE\_RANGE\_E](#_年龄段枚举). |
| udwSleevesLength | Sleeve length. See [NETDEV\_SLEEVES\_LENGTH\_E](#_上衣长短款式枚举) |
| udwCoatColor | Coat color. See [NETDEV\_CLOTHES\_COLOR\_E](#_衣服颜色枚举). |
| udwTrousersLength | Trousers length. See [NETDEV\_TROUSERS\_LENGTH\_E](#_下衣长短款式枚举) |
| udwTrousersColor | Trousers color. See [NETDEV\_CLOTHES\_COLOR\_E](#_衣服颜色枚举). |
| udwBodyToward | Body direction. See [NETDEV\_BODY\_TOWARD\_E](#_身体抓拍朝向枚举). |
| udwShoesTubeLength | Shoes tube length. See [NETDEV\_SHOES\_TUBE\_LENGTH\_E](#_鞋子长短款式枚举) |
| udwHairLength | Hair length. See [NETDEV\_HAIR\_LENGTH\_E](#_发型长短枚举). |
| udwBagFlag | Whether carrying a bag. See [NETDEV\_BAG\_FLAG\_E](#_是否携包标志枚举). |
| fTemperature | Body temperature (unit: °C, 2 decimal places) |
| udwMask | Mask. See [NETDEV\_PERSON\_MASK\_FLAG\_E](#_是否戴口罩枚举_1). |
| udwCoatTexture | Coat texture. See [NETDEV\_CLOTHES\_TEXTURE\_E](#_上衣纹理枚举). |
| udwMovingDirection | Moving direction of people. See [NETDEV\_MOVE\_DIRECTION\_E](#_人员运动方向枚举). |
| byRes | Reserved field. |

### Structured alarm info

|  |
| --- |
| typedef struct tagNETDEVStructAlarmInfo  {  CHAR szReference[NETDEV\_LEN\_260];  UINT32 udwTimeStamp;  UINT32 udwSeq;  UINT32 udwSrcID;  CHAR szSrcName[NETDEV\_LEN\_260];  UINT32 udwNotificationType;  CHAR szDeviceID[NETDEV\_LEN\_32];  CHAR szRelatedID[NETDEV\_LEN\_64];  BYTE byRes[32];  }NETDEV\_STRUCT\_ALARM\_INFO\_S,\*LPNETDEV\_STRUCT\_ALARM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szReference | Descriptive info |
| udwTimeStamp | Alarm time. Number of seconds since 0’o clock, Jan. 1, 1970 |
| udwSeq | Alarm sequence number |
| udwSrcID | Alarm source ID |
| szSrcName | Alarm source name |
| udwNotificationType | Notification type. 0: real-time notification 1: historical notification |
| szDeviceID | Alarm device ID. Filled in with the device’s GB code if the device is connected via GB. Length range: [1,32] |
| szRelatedID | ID that relates an alarm with alarm data (such as photos, attributes) or relates data of multiple channels. An ID is unique within a camera. Length: 15 characters. |
| byRes | Reserved field. |

### Structured data info

|  |
| --- |
| typedef struct tagNETDEVStructDataInfo  {  [NETDEV\_OBJECT\_INFO\_S](#_目标信息) stObjectInfo;  UINT32 udwImageNum;  [LPNETDEV\_STRUCT\_IMAGE\_INFO\_S](#_图像相关信息结构体) pstImageInfo;  BYTE byRes[128];  }NETDEV\_STRUCT\_DATA\_INFO\_S,\*LPNETDEV\_STRUCT\_DATA\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stObjectInfo | Object info |
| udwImageNum | Number of images |
| pstImageInfo | Image info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Object info

|  |
| --- |
| typedef struct tagNETDEVObjectInfo  {  UINT32 udwFaceNum;  [LPNETDEV\_FACE\_STRUCT\_INFO\_S](#_人脸信息)  pstFaceInfo;  UINT32 udwPersonNum;  [LPNETDEV\_PERSON\_STRUCT\_INFO\_S](#_人员信息) pstPersonInfo;  UINT32 udwNonMotorVehNum;  [LPNETDEV\_NON\_MOTOR\_VEH\_INFO\_S](#_非机动车信息) pstNonMotorVehInfo;  UINT32 udwVehicleNum;  [LPNETDEV\_VEH\_INFO\_S](#_车辆信息) pstVehInfo;  BYTE byRes[128];  }NETDEV\_OBJECT\_INFO\_S,\*LPNETDEV\_OBJECT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwFaceNum | Number of faces |
| pstFaceInfo | Face info. Memory needs to be allocated dynamically. |
| udwPersonNum | Number of persons |
| pstPersonInfo | Person info. Memory needs to be allocated dynamically. |
| udwNonMotorVehNum | Number of non-motor vehicles |
| pstNonMotorVehInfo | Non-motor vehicle info. Memory needs to be allocated dynamically. |
| udwVehicleNum | Number of vehicles |
| pstVehInfo | Vehicle Info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Face info

|  |
| --- |
| typedef struct tagNETDEVFaceStructInfo  {  UINT32 udwFaceID;  UINT32 udwFaceDoforPersonID;  CHAR szPosition[NETDEV\_LEN\_64];  UINT32 udwSmallPicAttachIndex;  UINT32 udwLargePicAttachIndex;  CHAR szFeaturVersion[NETDEV\_LEN\_64];  CHAR szFeature[NETDEV\_LEN\_1024];  [NETDEV\_FACE\_ATTR\_S](#_人脸属性信息) stFaceAttr;  [LPNETDEV\_RULE\_INFO\_S](#_规则信息) pstRuleInfo;  UINT32 udwFaceDoforNonMotorID;  UINT32 udwFaceDoforVehicleID;  BYTE byRes[116];  }NETDEV\_FACE\_STRUCT\_INFO\_S,\*LPNETDEV\_FACE\_STRUCT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwFaceID | Face ID |
| udwFaceDoforPersonID | ID of the corresponding person |
| szPosition | Face position info |
| udwSmallPicAttachIndex | Index of the small image of the face in the image list. |
| udwLargePicAttachIndex | Index of the corresponding complete image of the face in the image list. |
| szFeaturVersion | Version number of the algorithm extracting semi-structured attributes |
| szFeature | Extracted face attribute info. Base64-encoded. 512 bytes before encryption. |
| stFaceAttr | Face attribute info |
| pstRuleInfo | Rule info. Memory needs to be allocated dynamically. |
| udwFaceDoforNonMotorID | ID of the corresponding non-motor vehicle |
| udwFaceDoforVehicleID | ID of the corresponding motor vehicle |
| byRes | Reserved field. |

**See also:**

[NETDEV\_OBJECT\_INFO\_S](#_目标信息)

### Person info

|  |
| --- |
| typedef struct tagNETDEVPersonStructInfo  {  UINT32 udwPersonID;  UINT32 udwPersonDoforFaceID;  CHAR szPosition[NETDEV\_LEN\_64];  UINT32 udwSmallPicAttachIndex;  UINT32 udwLargePicAttachIndex;  CHAR szFeaturVersion[NETDEV\_LEN\_64];  CHAR szFeature[NETDEV\_LEN\_1024];  [NETDEV\_PERSON\_ATTR\_S](#_人员属性) stPersonAttr;  [LPNETDEV\_RULE\_INFO\_S](#_规则信息) pstRuleInfo;  BYTE byRes[124];  }NETDEV\_PERSON\_STRUCT\_INFO\_S,\*LPNETDEV\_PERSON\_STRUCT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPersonID | Person ID |
| udwPersonDoforFaceID | ID of the corresponding face |
| szPosition | Person position info |
| udwSmallPicAttachIndex | Index of the small image of the person in the image list |
| udwLargePicAttachIndex | Index of the corresponding complete image of the person in the image list |
| szFeaturVersion | Version number of the algorithm extracting semi-structured attributes |
| szFeature | Semi-structured attribute info. Base64-encoded. 512 bytes before encryption. |
| stPersonAttr | Person info |
| pstRuleInfo | Rule info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_OBJECT\_INFO\_S](#_目标信息)

### Non-motor vehicle info

|  |
| --- |
| typedef struct tagNETDEVNonMotorVehInfo  {  UINT32 udwID;  CHAR szPosition[NETDEV\_LEN\_64];  UINT32 udwSmallPicAttachIndex;  UINT32 udwLargePicAttachIndex;  [NETDEV\_NO\_MOTOR\_VEH\_ATTR\_S](#_非机动车属性信息) stNoMotorVehAttr;  UINT32 udwPersonOnNoVehiNum;  [LPNETDEV\_PERSON\_ATTR\_S](#_人员属性) pstPersonAttr;  [LPNETDEV\_RULE\_INFO\_S](#_规则信息) pstRuleInfo;  BYTE byRes[124];  }NETDEV\_NON\_MOTOR\_VEH\_INFO\_S,\*LPNETDEV\_NON\_MOTOR\_VEH\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Non-motor vehicle ID |
| szPosition | Non-motor vehicle position info |
| udwSmallPicAttachIndex | Index of the small image of the non-motor vehicle in the image list |
| udwLargePicAttachIndex | Index of the corresponding complete image of the non-motor vehicle in the image list |
| stNoMotorVehAttr | Non-motor vehicle attribute info |
| udwPersonOnNoVehiNum | Number of people in the vehicle |
| pstPersonAttr | Person attribute. Memory needs to be allocated dynamically. |
| pstRuleInfo | Rule info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_OBJECT\_INFO\_S](#_目标信息)

### Non-motor vehicle attribute info

|  |
| --- |
| typedef struct tagNETDEVNonMotorVehAttr  {  UINT32 udwSpeedType;  UINT32 udwImageDirection;  UINT32 udwNonVehicleType ;  BYTE byRes[128];  }NETDEV\_NO\_MOTOR\_VEH\_ATTR\_S,\*LPNETDEV\_NO\_MOTOR\_VEH\_ATTR\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSpeedType | Speed type of non-motor vehicle in structured scenes. See [NETDEV\_SPEED\_TYPE\_E](#_结构化场景中非机动车速度类型枚举). |
| udwImageDirection | Driving direction (relative to the image) of non-motor vehicle in structured scenes.  See [NETDEV\_IMAGE\_DIRECTION\_E](#_结构化场景中非机动车相对画面运动方向枚举). |
| udwNonVehicleType | Non-motor vehicle type. See [NETDEV\_NON\_VEH\_TYPE\_E](#_非机动车类型枚举). |
| byRes | Reserved field. |

### Vehicle info

|  |
| --- |
| typedef struct tagNETDEVVehicleInfo  {  UINT32 udwID;  CHAR szPosition[NETDEV\_LEN\_64];  UINT32 udwSmallPicAttachIndex ;  UINT32 udwLargePicAttachIndex;  UINT32 udwPlatePicAttachIndex;  CHAR szFeatureVersion[NETDEV\_LEN\_64];  CHAR szFeature[NETDEV\_LEN\_1024];  [NETDEV\_VEH\_ATTR\_S](#_车辆属性信息) stVehAttr;  [NETDEV\_PLATE\_ATTR\_S](#_车牌属性信息) stPlateAttr;  [LPNETDEV\_RULE\_INFO\_S](#_规则信息) pstRuleInfo;  BYTE byRes[124];  }NETDEV\_VEH\_INFO\_S, \*LPNETDEV\_VEH\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Vehicle ID |
| szPosition | Vehicle location info |
| udwSmallPicAttachIndex | Index of the small image of the vehicle in the image list |
| udwLargePicAttachIndex | Index of the corresponding complete image of the vehicle in the image list |
| udwPlatePicAttachIndex | Index of the small image of the plate in the image list |
| szFeatureVersion | Version number of the algorithm extracting semi-structured attributes |
| szFeature | Extracted face attribute info. Base64-encoded. 512 bytes before encryption. |
| stVehAttr | Vehicle attribute info |
| stPlateAttr | Plate attribute info |
| pstRuleInfo | Rule info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_OBJECT\_INFO\_S](#_目标信息)

### Vehicle attribute info

|  |
| --- |
| typedef struct tagNETDEVVehAttr  {  UINT32 udwType;  UINT32 udwColor;  UINT32 udwSpeedUnit;  FLOAT fSpeedValue;  UINT32 udwSpeedType;  CHAR szVehicleBrand[NETDEV\_LEN\_64];  UINT32 udwImageDirection;  BYTE byRes[128];  }NETDEV\_VEH\_ATTR\_S,\*LPNETDEV\_VEH\_ATTR\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | Vehicle type. See [NETDEV\_VEHICLE\_TYPE\_E](#_车辆类型枚举). |
| udwColor | Vehicle color. See [NETDEV\_PLATE\_COLOR\_E](#_车牌颜色枚举). |
| udwSpeedUnit | Vehicle speed unit. 0: kph 1: mph |
| fSpeedValue | Vehicle speed |
| udwSpeedType | Vehicle speed type in structured scenes. See [NETDEV\_SPEED\_TYPE\_E](#_结构化场景中非机动车速度类型枚举). |
| szVehicleBrand | Vehicle brand (custom) |
| udwImageDirection | Driving direction (relative to the image) of motor vehicle in structured scenes. See  [NETDEV\_IMAGE\_DIRECTION\_E](#_结构化场景中非机动车相对画面运动方向枚举) |
| byRes | Reserved field. |

### Plate attribute info

|  |
| --- |
| typedef struct tagNETDEVPlateAttr  {  CHAR szPlateNo[NETDEV\_LEN\_64];  UINT32 udwColor;  UINT32 udwType;  BYTE byRes[128];  }NETDEV\_PLATE\_ATTR\_S,\*LPNETDEV\_PLATE\_ATTR\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szPlateNo | Plate No. |
| udwColor | Plate color. See [NETDEV\_PLATE\_COLOR\_E](#_车牌颜色枚举). |
| udwType | Plate type. See [NETDEV\_PLATE\_TYPE\_E](#_车牌类型枚举). |
| byRes | Reserved field. |

### Structure of vehicle recognition event

|  |
| --- |
| typedef struct tagstNETDEVVehRecognitionEvent  {  CHAR szReference[NETDEV\_LEN\_480];  UINT32 udwSrcID;  [NETDEV\_VEHICLE\_EVENT\_INFO\_S](#_车辆比对报警信息结构体) stVehicleEventInfo;  BYTE byRes[256];  }NETDEV\_VEH\_RECOGNITION\_EVENT\_S, \*LPNETDEV\_VEH\_RECOGNITION\_EVENT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szReference | Subscriber info |
| udwSrcID | Alarm source ID |
| stVehicleEventInfo | Vehicle comparison alarm info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_VehicleAlarmMessCallBack\_PF](#_注册车牌识别报警回调函数)

### Structure of vehicle comparison alarm info

|  |
| --- |
| typedef struct tagstVehicleEventInfo  {  UINT32 udwID;  UINT32 udwTimestamp;  UINT32 udwNotificationType;  UINT32 udwVehicleInfoNum;  [LPNETDEV\_VEHICLE\_RECORD\_INFO\_S](#_车辆识别记录信息结构体) pstVehicleRecordInfo;  BYTE byRes[256];  }NETDEV\_VEHICLE\_EVENT\_INFO\_S,\*LPNETDEV\_VEHICLE\_EVENT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Notification record ID |
| udwTimestamp | Notification reporting time (unit: second, UTC format). |
| udwNotificationType | Notification type. See [NETDEV\_NOTIFICATION\_TYPE\_E](#_通知类型枚举). |
| udwVehicleInfoNum | Number of vehicle info: [0, 1] |
| pstVehicleRecordInfo | Vehicle list |
| byRes | Reserved field. |

### Structure of vehicle recognition record info

|  |
| --- |
| typedef struct tagstNETDEVVehicleRcordInfo  {  UINT32 udwRecordID;  UINT32 udwChannelID;  UINT32 udwPassingTime;  CHAR szChannelName[NETDEV\_LEN\_260];  [NETDEV\_PLATE\_ATTR\_INFO\_S](#_车牌信息结构体) stPlateAttr;  [NETDEV\_VEH\_ATTR\_S](#_车辆属性信息) stVehAttr;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stPlateImage;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stVehicleImage;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stPanoImage;  [NETDEV\_MONITOR\_ALARM\_INFO\_S](#_车牌告警布控信息结构体) stMonitorAlarmInfo;  BYTE byRes[132];  }NETDEV\_VEHICLE\_RECORD\_INFO\_S, \*LPNETDEV\_VEHICLE\_RECORD\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwRecordID | Vehicle recognition record ID |
| udwChannelID | Capture channel ID, valid when pushing captured images. |
| udwPassingTime | Vehicle pass-through time (unit: second, UTC format). |
| szChannelName | Checkpoint camera name |
| stPlateAttr | Captured plate info |
| stVehAttr | Captured vehicle info |
| stPlateImage | Captured plate image. Max 1MB after encryption. |
| stVehicleImage | Captured vehicle image. Required in structured queries. Max 1MB after encryption. |
| stPanoImage | Panoramic image. Required in structured queries. Only image URL and size are included. Image data are obtained through /LAPI/V1.0/System/Picture |
| stMonitorAlarmInfo | Plate monitoring info |
| byRes | Reserved field. |

### Structure of plate info

|  |
| --- |
| typedef struct tagstNETDEVPlateAttrInfo  {  CHAR szPlateNo[NETDEV\_LEN\_16];  UINT32 udwColor;  UINT32 udwType;  BYTE byRes[64];  }NETDEV\_PLATE\_ATTR\_INFO\_S, \*LPNETDEV\_PLATE\_ATTR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szPlateNo | Plate No. |
| udwColor | Plate color. See [NETDEV\_PLATE\_COLOR\_E](#_车牌颜色枚举). |
| udwType | Plate type. See [NETDEV\_PLATE\_TYPE\_E](#_车牌类型枚举). |
| byRes | Reserved field. |

### Structure of info about smart event subscription

|  |
| --- |
| typedef struct tagstNETDEVSubscribeSmartInfo  {  UINT32 udwNum;  UINT32 \*pudwSmartType;  BYTE byRes[128];  }NETDEV\_SUBSCRIBE\_SMART\_INFO\_S, \*LPNETDEV\_SUBSCRIBE\_SMART\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of subscribed smart alarms |
| pudwSmartType | Subscribed smart alarm types. See [NETDEV\_SMART\_ALARM\_TYPE\_E](#_智能告警类型枚举). Memory needs to be allocated dynamically according to udwNum. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SubscribeSmart](#_订阅智能事件)

### Structure of smart event info

|  |
| --- |
| typedef struct tagstNETDEVSmartInfo  {  INT32 dwChannelID;  UINT32 udwSubscribeID;  UINT32 udwSubscribeType;  UINT32 udwCurrrntTime;  UINT32 udwEndTime;  BYTE byRes[124];  }NETDEV\_SMART\_INFO\_S, \*LPNETDEV\_SMART\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| udwSubscribeID | Subscription ID |
| udwSubscribeType | Subscription type |
| udwCurrrntTime | Current time (unit: second, UTC format). |
| udwEndTime | End time (unit: second, UTC format). |
| byRes | Reserved field. |

**See also:**

[NETDEV\_SubscribeSmart](#_订阅智能事件), [NETDEV\_UnsubscribeSmart](#_取消订阅智能事件)

### Structure of info about LAPI alarm subscription

|  |
| --- |
| typedef struct tagNETDEVLapiSubInfo  {  UINT32 udwType;  UINT32 udwLibIDNum;  UINT32 audwLibIDList[NETDEV\_LEN\_32];  BYTE byRes[132];  }NETDEV\_LAPI\_SUB\_INFO\_S,\*LPNETDEV\_LAPI\_SUB\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | Subscription type. Indicated by bit. See [NETDEV\_ALARM\_TYPE\_V30\_E](#_告警支持类型枚举). |
| udwLibIDNum | Number of subscribed library IDs. LibIDNum=0xffff means subscribing to all libraries |
| audwLibIDList | List of IDs of subscribed libraries. |
| byRes | Reserved field |

**See also:**

[NETDEV\_SubscibeLapiAlarm](#_LAPI告警订阅)

### Structure of returned result for successful subscriptions

|  |
| --- |
| typedef struct tagNETDEVSubscribeSuccInfo  {  UINT32 udwID;  UINT32 udwCurrrntTime;  UINT32 udwTerminationTime;  UINT32 udwSupportAlarmType;  CHAR szReference[NETDEV\_LEN\_260];  BYTE byRes[124];  }NETDEV\_SUBSCRIBE\_SUCC\_INFO\_S,\*LPNETDEV\_SUBSCRIBE\_SUCC\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Subscription ID |
| udwCurrrntTime | Current time (UTC format). Number of seconds since 0 o’clock Jan. 1, 1970 |
| udwTerminationTime | End time (UTC format). Number of seconds since 0 o’clock Jan. 1, 1970 |
| udwSupportAlarmType | The return value shall carry this parameter if the request message carries the subscribed alarm type. 0 means the response does not carry the data. |
| szReference | Subscriber info. In the form of a URL. |
| byRes | Reserved field |

**See also:**

[NETDEV\_SubscibeLapiAlarm](#_LAPI告警订阅)

### Structure of returned alarm records (face recognition and plate recognition)

|  |
| --- |
| typedef struct tagstNETDEVSmartAlarmLogResultInfo  {  UINT32 udwTotal;  UINT32 udwOffset;  UINT32 udwNum;  BYTE byRes[128];  }NETDEV\_SMART\_ALARM\_LOG\_RESULT\_INFO\_S, \*LPNETDEV\_SMART\_ALARM\_LOG\_RESULT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTotal | Total number of alarm records |
| udwOffset | Record offset |
| udwNum | Number of alarm records returns this time |
| byRes | Reserved field. |

**See also:**

[[NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录)](#_查询人脸识别记录)

### Structure of parking lot entrance/exit list

|  |
| --- |
| typedef struct tagstNETDEVParkEntranceInfoList  {  UINT32 udwNum;  [NETDEV\_PARK\_ENTRANCE\_INFO\_S](#_停车场出入口信息结构体) astParkEntranceInfo[NETDEV\_LEN\_32];  BYTE byRes[512];  }NETDEV\_PARK\_ENTRANCE\_INFO\_LIST\_S,\*LPNETDEV\_PARK\_ENTRANCE\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of entrances/exits |
| astParkEntranceInfo | Entrance/exit info. Each parking lot supports up to 16 entrances/exits. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_GetParkEntranceInfoList](#_获取某停车场下所有出入口信息)

### Structure of parking lot entrance/exit info

|  |
| --- |
| typedef struct tagstNETDEVParkEntranceInfo  {  UINT32 udwParkEntranceID;  CHAR szName[NETDEV\_LEN\_260];  UINT32 udwVehicleLaneNum;  BYTE byRes[256];  }NETDEV\_PARK\_ENTRANCE\_INFO\_S, \*LPNETDEV\_PARK\_ENTRANCE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwParkEntranceID | Entrance/exit ID. Not carried when adding an entrance/exit |
| szName | Entrance/exit name. Length range: [1,64] |
| udwVehicleLaneNum | Number of lanes. Max: 2 |
| byRes | Reserved field. |

### Structure of parking lot list

|  |
| --- |
| typedef struct tagstNETDEVParkInfoList  {  UINT32 udwNum;  [NETDEV\_PARK\_INFO\_S](#_停车场信息结构体) astParkInfo[NETDEV\_LEN\_16];  BYTE byRes[512];  }NETDEV\_PARK\_INFO\_LIST\_S, \*LPNETDEV\_PARK\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of parking lots |
| astParkInfo | Parking lot info. Each VMS allows up to 8 parking lots. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_GetParkInfoList](#_批量获取停车场信息)

### Structure of parking log info

|  |
| --- |
| typedef struct tagstNETDEVParkInfo  {  UINT32 udwParkingLotID;  CHAR szParkingLotName[NETDEV\_LEN\_260];  UINT32 udwTotalSpaceNum;  UINT32 udwRmnSpaceNum;  UINT32 udwReserveSpaceNum;  UINT32 udwRmnReserveSpaceNum;  UINT32 udwMainParkingLotID;  BYTE byRes[256];  }NETDEV\_PARK\_INFO\_S,\*LPNETDEV\_PARK\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwParkingLotID | Parking lot ID |
| szParkingLotName | Parking lot name, length range: [1,64]. |
| udwTotalSpaceNum | Total number of parking spaces |
| udwRmnSpaceNum | Number of remaining parking spaces |
| udwReserveSpaceNum | Number of fixed parking spaces |
| udwRmnReserveSpaceNum | Number of remaining fixed parking spaces |
| udwMainParkingLotID | Parking lot ID. Optional in the absence of main parking lot. |
| byRes | Reserved field. |

### Structure of parking lot entrance/exit ID

|  |
| --- |
| typedef struct tagstNETDEVParkEntranceID  {  UINT32 udwParkID;  UINT32 udwEntranceID;  BYTE byRes[256];  }NETDEV\_PARK\_ENTRANCE\_ID\_S,\*LPNETDEV\_PARK\_ENTRANCE\_ID\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwParkID | Parking lot ID |
| udwEntranceID | Entrance/exit ID |
| byRes | Reserved field. |

**See also:**

[NETDEV\_GetParkLaneInfoList](#_获取停车场出入口所有车道信息)

### Structure of lane list

|  |
| --- |
| typedef struct tagstNETDEVVehLaneInfoList  {  UINT32 udwNum;  [NETDEV\_VEH\_LANE\_INFO\_S](#_车道信息结构体) astVehLaneInfo[NETDEV\_LEN\_4];  BYTE byRes[128];  }NETDEV\_VEH\_LANE\_INFO\_LIST\_S,\*LPNETDEV\_VEH\_LANE\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of lanes. Max: 2 |
| astVehLaneInfo | Lane information |
| byRes | Reserved field. |

**See also:**

[NETDEV\_GetParkLaneInfoList](#_获取停车场出入口所有车道信息)

### Structure of lane info

|  |
| --- |
| typedef struct tagstNETDEVVehLaneInfo  {  UINT32 udwVehLaneID;  CHAR szName[NETDEV\_LEN\_128];  UINT32 udwType;  UINT32 udwNum;  [NETDEV\_TM\_SECT\_INFO\_S](#_自定义时间段结构体) astEnTmSectInfo[NETDEV\_LEN\_4];  [NETDEV\_VEH\_THROUGH\_INFO\_S](#_车辆放行信息结构体) stTmpVehThrough;  [NETDEV\_VEH\_THROUGH\_INFO\_S](#_车辆放行信息结构体) stRegVehThrough;  UINT32 udwDevNum;  UINT32 audwDevID[NETDEV\_LEN\_4];  BYTE byRes[128];  }NETDEV\_VEH\_LANE\_INFO\_S,\*LPNETDEV\_VEH\_LANE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwVehLaneID | Lane ID |
| szName | Lane Name |
| udwType | Lane type. See [NETDEV\_VEH\_LANE\_TYPE\_E](#_车道类型枚举). |
| udwNum | Number of custom periods. Max: 4. |
| astEnTmSectInfo | Custom period list |
| stTmpVehThrough | Let through temporary vehicles |
| stRegVehThrough | Let through regular vehicles |
| udwDevNum | Number of devices. Max: 4 |
| audwDevID | List of IDs of entrance/exit devices linked to lanes |
| byRes | Reserved field. |

### Structure of custom time periods

|  |
| --- |
| typedef struct tagstNETDEVTmSectInfo  {  CHAR szBegin[NETDEV\_LEN\_64];  CHAR szEnd[NETDEV\_LEN\_64];  BYTE byRes[128];  }NETDEV\_TM\_SECT\_INFO\_S,\*LPNETDEV\_TM\_SECT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szBegin | Start time (hh:mm:ss), length range: [0,31] |
| szEnd | End time (hh:mm:ss), length range: [0,31] |
| byRes | Reserved field. |

### Structure of vehicle let-through info

|  |
| --- |
| typedef struct tagstNETDEVVehThroughInfo  {  UINT32 udwType;  UINT32 udwNum;  [NETDEV\_TM\_SECT\_INFO\_S](#_自定义时间段结构体) astTmSectInfo[NETDEV\_LEN\_4];  BYTE byRes[128];  }NETDEV\_VEH\_THROUGH\_INFO\_S,\*LPNETDEV\_VEH\_THROUGH\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | Release method. See [NETDEV\_VEH\_RELEASE\_TYPE\_E](#_车辆放行类型枚举). |
| udwNum | Custom let-through period list. Max: 4 |
| astTmSectInfo | Custom let-through time periods |
| byRes | Reserved field. |

### Structure of parking payment order info

|  |
| --- |
| typedef struct tagstNETDEVParkPayOrderInfo  {  CHAR szOrderNo[NETDEV\_LEN\_128];  UINT32 udwOrderStatus;  CHAR szPlateNo[NETDEV\_LEN\_128];  UINT32 udwPayType;  CHAR szPayTradeNo[NETDEV\_LEN\_128];  UINT32 udwTotalAmount;  UINT32 udwDiscountAmount;  BYTE byRes[256];  }NETDEV\_PARK\_PAY\_ORDER\_INFO\_S,\*LPNETDEV\_PARK\_PAY\_ORDER\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szOrderNo | Order number, generated by the third-party |
| udwOrderStatus | Order payment status. See [NETDEV\_ORDER\_STATUS\_E](#_订单支付状态枚举). |
| szPlateNo | Plate No. |
| udwPayType | Payment type. See [NETDEV\_PAY\_TYPE\_E](#_支付类型枚举). |
| szPayTradeNo | Payment order number |
| udwTotalAmount | Total amount of parking fees (unit: cent) |
| udwDiscountAmount | Discounted amount (unit: cent) |
| byRes | Reserved field. |

**See also:**

[NETDEV\_IssuePayOrder](#_下发第三方支付订单)

### Structure of parking cost

|  |
| --- |
| typedef struct tagstNETDEVParkCost  {  UINT32 udwTotalAmount;  UINT32 udwEnterTime;  UINT32 udwExitLimitTime;  BYTE byRes[256];  }NETDEV\_PARK\_COST\_S, \*LPNETDEV\_PARK\_COST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTotalAmount | Discounted amount (unit: cent) |
| udwEnterTime | Entry time (unit: second, UTC format) |
| udwExitLimitTime | Specified time limit for leaving the parking lot (unit: minute) |
| byRes | Reserved field. |

**See also:**

[NETDEV\_IssuePayOrder](#_下发第三方支付订单)

### Structure of parking event info

|  |
| --- |
| typedef struct tagstNETDEVParkEventInfo  {  UINT32 udwID;  UINT32 udwParkID;  CHAR szPlateNo[NETDEV\_LEN\_32];  [NETDEV\_VEH\_ATTR\_S](#_车辆属性信息) stVehAttr;  UINT32 udwVehRemainNum;  UINT32 udwIsWhiteList;  UINT32 udwIsBlackList;  CHAR szEntranceName[NETDEV\_LEN\_128];  CHAR szLaneName[NETDEV\_LEN\_128];  UINT32 udwCrossDirection;  [LPNETDEV\_ENTR\_INFO\_S](#_入场车辆抓拍信息结构体) pstEnterInfo;  [LPNETDEV\_EXIT\_INFO\_S](#_出场车辆抓拍信息结构体) pstExitInfo;  BYTE byRes[256];  }NETDEV\_PARK\_EVENT\_INFO\_S,\*LPNETDEV\_PARK\_EVENT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Record ID |
| udwParkID | Parking lot ID |
| szPlateNo | Plate No. |
| stVehAttr | Vehicle attribute info |
| udwVehRemainNum | Number of vehicles in the parking lot |
| udwIsWhiteList | Whether is on the allowlist. 0: No 1: Yes |
| udwIsBlackList | Whether is on the blocklist. 0: No 1: Yes |
| szEntranceName | Entrance & exit name |
| szLaneName | Lane name |
| udwCrossDirection | Vehicle direction. 0: Enter 1: Leave |
| pstEnterInfo | Entry information |
| pstExitInfo | Exit information |
| byRes | Reserved field. |

**See also:**

[NETDEV\_ParkEventMessCallBack\_PF](#_注册停车场事件回调函数)

### Structure of vehicle entry image

|  |
| --- |
| typedef struct tagNETDEVEntrInfo  {  UINT32 udwEnterTime;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stPlateImage;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stVehImage;  BYTE byRes[256];  }NETDEV\_ENTR\_INFO\_S,\*LPNETDEV\_ENTR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwEnterTime | Entry time (unit: second, UTC format) |
| stPlateImage | Plate image |
| stVehImage | Vehicle image |

### Structure of vehicle leaving image

|  |
| --- |
| typedef struct tagstNETDEVExitInfo  {  UINT32 udwExitTime;  UINT32 udwTotalAmount;  UINT32 udwIsAlreadyPaid;  UINT32 udwChargeRecordID;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stPlateImage;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stVehImage;  CHAR szOperatorName[NETDEV\_LEN\_128];  BYTE byRes[256];  }NETDEV\_EXIT\_INFO\_S,\*LPNETDEV\_EXIT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwExitTime | Exit time (unit: second, UTC format) |
| udwTotalAmount | Parking fees (unit: cent) |
| udwIsAlreadyPaid | Whether parking fees are paid. 0: No 1: Yes |
| udwChargeRecordID | Payment record ID |
| stPlateImage | Plate image |
| stVehImage | Vehicle image |
| szOperatorName | Operator |
| byRes | Reserved field. |

### Structure of time template configuration

|  |
| --- |
| typedef struct tagNETDEVSystemTimeTemplate  {  UINT32 udwTemplateID;  CHAR szTemplateName[NETDEV\_LEN\_256];  CHAR szTemplateDesc[NETDEV\_LEN\_512];  UINT32 udwLastChange;  [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) stWeekPlanInfo;  [NETDEV\_EXCEPTION\_INFO\_S](#_布控任务例外计划结构体) stExceptionInfo;  BOOL bIsBuiltin;  UINT32 udwTemplateType;  BYTE byRes[128];  }NETDEV\_SYSTEM\_TIME\_TEMPLATE\_S,\*LPNETDEV\_SYSTEM\_TIME\_TEMPLATE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTemplateID | Time template ID |
| szTemplateName | Time template name. Range: [1, 63] |
| szTemplateDesc | Time template description. Range: [1, 128] |
| udwLastChange | Last time time template is modified. |
| stWeekPlanInfo | Arming scheduling of monitoring task |
| stExceptionInfo | Exception schedule of monitoring task |
| bIsBuiltin | Whether is a built-in time template. 1: Yes 2: No. Applicable to VMS only. |
| udwTemplateType | Time template type. Applicable to VMS. 0: Recording time template, 1: Alarm time template, 2: User time template. |
| byRes | Reserved field |

**See also:**

[NETDEV\_AddTimeTemplate](#_添加时间模板), [NETDEV\_GetTimeTemplate](#_获取指定时间模板信息), [NETDEV\_ModifyTimeTemplate](#_修改指定时间模板信息)

### Structure of exceptions of monitoring task schedule

|  |
| --- |
| typedef struct tagNETDEVExceptionInfo  {  BOOL bEnabled;  UINT32 udwNum;  [NETDEV\_EXCEPTION\_DAY\_INFO\_S](#_每天的布防计划具体信息结构体) stExceptionDayInfo[NETDEV\_LEN\_32];  BYTE byRes[128];  }NETDEV\_EXCEPTION\_INFO\_S,\*LPNETDEV\_EXCEPTION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Whether exception dates are enabled. 0: No 1: Yes. |
| udwNum | Number of exception dates. Range: [0, 16]. |
| stExceptionDayInfo | Detailed arming schedule information of each day |
| byRes | Reserved field |

### Structure of detailed info about daily arming schedule

|  |
| --- |
| typedef struct tagNETDEVExceptionDayInfo  {  UINT32 udwID;  BOOL bEnabled;  CHAR szDate[NETDEV\_LEN\_16];  UINT32 udwNum;  [NETDEV\_TIME\_SECTION\_INFO\_S](#_时间段配置) stTimeSectionInfo[NETDEV\_LEN\_16];  BYTE byRes[128];  }NETDEV\_EXCEPTION\_DAY\_INFO\_S,\*LPNETDEV\_EXCEPTION\_DAY\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Exception date index |
| bEnabled | Whether exception dates are enabled |
| szDate | Exception dates. |
| udwNum | Number of exception periods. NVR: Max 8. IPC/PTS: Max 4. |
| stTimeSectionInfo | Detailed arming configuration information |
| byRes | Reserved field |

### Structure of general-purpose ID list

|  |
| --- |
| typedef struct tagNETDEVIDList  {  UINT32 udwNum;  UINT32 \*pudwIDs;  BYTE byRes[256];  }NETDEV\_ID\_LIST\_S, \*LPNETDEV\_ID\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Quantity |
| pudwIDs | ID list. Memory needs to be allocated dynamically by Malloc. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_BatchDeleteTimeTemplate](#_批量删除指定的时间模板)

### Structure of time template

|  |
| --- |
| typedef struct tagNETDEVTimeTemplateBaseInfo  {  UINT32 udwTemplateID;  CHAR szTemplateName[NETDEV\_LEN\_256];  UINT32 udwLastChange;  BYTE byRes[256];  }NETDEV\_TIME\_TEMPLATE\_BASE\_INFO\_S, \*LPNETDEV\_TIME\_TEMPLATE\_BASE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTemplateID | Template ID |
| szTemplateName | Template name |
| udwLastChange | Last time modified. |
| byRes | Reserved field. |

**See also:**

[[NETDEV\_FindNextTimeTemplateByTypeInfo](#_逐个获取按类型找到的时间模板信息)](#_逐个获取按类型找到的时间模板信息)

### Structure of access control person info

|  |
| --- |
| typedef struct tagNETDEVACSPersonInfo  {  UINT32 udwReqSeq;  UINT32 udwPersonID;  CHAR szName[NETDEV\_LEN\_260];  UINT32 udwGender;  [NETDEV\_FACE\_MEMBER\_ID\_INFO\_S](#_成员证件信息结构体_1) stMemberIDInfo;  CHAR szTelephone[NETDEV\_LEN\_64];  CHAR szAddress[NETDEV\_LEN\_260];  CHAR szDesc[NETDEV\_LEN\_480];  UINT32 udwCardNum;  [NETDEV\_ACS\_PERSON\_CARD\_INFO\_S](#_人员所持门禁卡信息结构体) stACSPersonCardList[NETDEV\_LEN\_6];  [NETDEV\_ACS\_FACE\_IMAGE\_S](#_图片信息结构体) stFaceImage;  UINT32 udwType;  [NETDEV\_ACS\_STAFF\_INFO\_S](#_员工信息结构体_1) stStaffInfo;  [NETDEV\_ACS\_VISITOR\_INFO\_S](#_访客信息结构体_1) stVisitor;  BYTE byRes[256];  }NETDEV\_ACS\_PERSON\_INFO\_S, \*LPNETDEV\_ACS\_PERSON\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwReqSeq | Sequence number of request |
| udwPersonID | Person ID |
| szName | Name |
| udwGender | Gender. See [NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举). |
| stMemberIDInfo | ID information. |
| szTelephone | Phone number |
| szAddress | Address |
| szDesc | Remarks |
| udwCardNum | Number of access control cards. Range: [1, 6]. |
| stACSPersonCardList | Access control card information |
| stFaceImage | Face image |
| udwType | Person type. See [NETDEV\_ACS\_PERSON\_TYPE\_E](#_人员类型枚举). |
| stStaffInfo | Staff info |
| stVisitor | Visitor information |
| byRes | Reserved field. |

**See also:**

[NETDEV\_ACSPersonCtrl](#_门禁人员管理)

### Structure of image info

|  |
| --- |
| typedef struct tagNETDEVACSFaceImage  {  UINT32 udwNum;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stImageList[NETDEV\_LEN\_16];  UINT32 udwMajorImageIndex;  BYTE byRes[128];  }NETDEV\_ACS\_FACE\_IMAGE\_S, \*LPNETDEV\_ACS\_FACE\_IMAGE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of images |
| stImageList | Face image list |
| udwMajorImageIndex | Major image index |
| byRes | Reserved field. |

### Structure of staff info

|  |
| --- |
| typedef struct tagNETDEVACSStaffInfo  {  CHAR szNumber[NETDEV\_LEN\_16];  CHAR szBirthday[NETDEV\_FACE\_MEMBER\_BIRTHDAY\_LEN];  CHAR szDeptName[NETDEV\_LEN\_260];  UINT32 udwDeptID;  BYTE byRes[128];  }NETDEV\_ACS\_STAFF\_INFO\_S, \*LPNETDEV\_ACS\_STAFF\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szNumber | Person ID |
| szBirthday | Birth date |
| szDeptName | Department name |
| udwDeptID | Department ID |

### Structure of time info

|  |
| --- |
| typedef struct tagACSTimeSection  {  INT64 tStartTime;  INT64 tEndTime;  BYTE byRes[32];  }NETDEV\_ACS\_TIME\_SECTION\_S, \*LPNETDEV\_ACS\_TIME\_SECTION\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| tStartTime | Start time (unit: second, UTC format) |
| tEndTime | End time (unit: second, UTC format) |
| byRes | Reserved field. |

### Structure of visitor info

|  |
| --- |
| typedef struct tagNETDEVACSVisitorInfo  {  CHAR szVisitorCompany[NETDEV\_LEN\_260];  UINT32 udwVisitorCount;  UINT32 udwIntervieweeID;  CHAR szIntervieweeName[NETDEV\_LEN\_260];  CHAR szIntervieweeDept[NETDEV\_LEN\_260];  [NETDEV\_ACS\_TIME\_SECTION\_S](#_时间信息结构体) tScheduleTime;  [NETDEV\_ACS\_TIME\_SECTION\_S](#_时间信息结构体) tRealTime;  UINT32 udwStatus;  BYTE byRes[128];  }NETDEV\_ACS\_VISITOR\_INFO\_S, \*LPNETDEV\_ACS\_VISITOR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szVisitorCompany | Company of the visitor. Range: [1,64] characters |
| udwVisitorCount | Number of visitors |
| udwIntervieweeID | ID of the visited |
| szIntervieweeName | Name of the visited. Range: [1,64] characters |
| szIntervieweeDept | Department of the visited. Range: [1,64] characters |
| tScheduleTime | Scheduled visit time |
| tRealTime | Actual visit time |
| udwStatus | Status. See [NETDEV\_ACS\_VISIT\_STATUS\_E](#_访客状态枚举). |
| byRes | Reserved field. |

### Structure of info about person’s access control card

|  |
| --- |
| typedef struct tagNETDEVACSPersonCardInfo  {  UINT32 udwCardID;  UINT32 udwCardType;  UINT32 udwCardStatus;  CHAR szCardNo[NETDEV\_LEN\_64];  UINT32 udwReqSeq;  [NETDEV\_ACS\_TIME\_SECTION\_S](#_时间信息结构体) stValidTime;  BYTE byRes[256];  }NETDEV\_ACS\_PERSON\_CARD\_INFO\_S,\* LPNETDEV\_ACS\_PERSON\_CARD\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwCardID | Bound ID |
| udwCardType | Card type |
| udwCardStatus | Card status. 0:Blank 1:Active 2: Frozen 3: Cancelled |
| szCardNo | Card number |
| udwReqSeq | Sequence number |
| stValidTime | Valid period |
| byRes | Reserved field. |

**See also:**

[NETDEV\_AddACSPersonCardInfo](#_绑定指定人员的门卡信息), [NETDEV\_GetACSPersonCardInfo](#_获取指定人员所持门禁卡信息)

### Structure of blocklist info

|  |
| --- |
| typedef struct tagNETDEVACSPersonBlacklistInfo  {  UINT32 udwBlackListID;  [NETDEV\_FACE\_MEMBER\_ID\_INFO\_S](#_成员证件信息结构体_1) stIdentificationInfo;  BYTE byRes[256];  }NETDEV\_ACS\_PERSON\_BLACKLIST\_INFO\_S, \*LPNETDEV\_ACS\_PERSON\_BLACKLIST\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwBlackListID | Blocklist ID |
| stIdentificationInfo | ID info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_AddACSPersonBlackList](#_添加访客黑名单), [NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录),

[NETDEV\_GetACSPersonBlackList](#_获取指定访客黑名单信息), [NETDEV\_ModifyACSPersonBlackList](#_修改访客黑名单信息)

### Structure of person list

|  |
| --- |
| typedef struct tagNETDEVACSPersonList  {  UINT32 udwNum;  [LPNETDEV\_ACS\_PERSON\_INFO\_S](#_门禁人员信息结构体) pstPersonInfoList;  BYTE byRes[128];  }NETDEV\_ACS\_PERSON\_LIST\_S, \*LPNETDEV\_ACS\_PERSON\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of persons |
| pstPersonInfoList | Staff list. Memory needs to be allocated dynamically according to udwNum. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_AddACSPersonList](#_批量添加人员信息)

### Structure of permission info

|  |
| --- |
| typedef struct tagNETDEVACSPermissionInfo  {  UINT32 udwPermissionID;  CHAR szPermissionName[NETDEV\_LEN\_260];  UINT32 udwPermissionType;  [NETDEV\_OPERATE\_LIST\_S](#_批处理列表) stPersonList;  UINT32 udwTemplateID;  [NETDEV\_ACS\_TIME\_SECTION\_S](#_时间信息结构体) stValidTime;  [NETDEV\_OPERATE\_LIST\_S](#_批处理列表) stDoorList;  BYTE byRes[256];  }NETDEV\_ACS\_PERMISSION\_INFO\_S, \*LPNETDEV\_ACS\_PERMISSION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPermissionID | Permission ID |
| szPermissionName | Permission name |
| udwPermissionType | Permission type. 0: Staff permission group 1: Visitor permission group |
| stPersonList | Person ID list, where dwSize is the number of persons. |
| udwTemplateID | Time template ID |
| stValidTime | Valid time |
| stDoorList | Access control channel list, where dwSize is the number of access control channels. |

**See also:**

[NETDEV\_AddACSPersonPermissionGroup](#_添加人员授权组信息), [NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1),

[NETDEV\_GetSinglePermGroupInfo](#_获取单个授权组信息), [NETDEV\_ModifyACSPersonPermissionGroup](#_修改人员授权组信息)

### Structure of batch operation list of face recognition module

|  |
| --- |
| typedef struct tagNETDEVFaceBatchList  {  UINT32 udwNum;  [LPNETDEV\_FACE\_BATCH\_INFO\_S](#_人脸识别模块批量操作信息结构体) pstBatchList;  BYTE byRes[128];  }NETDEV\_FACE\_BATCH\_LIST\_S,\*LPNETDEV\_FACE\_BATCH\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Batch processed number |
| pstBatchList | Batch operation info. Memory needs to be allocated dynamically according to udwNum. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_DeleteACSPersonList](#_批量删除门禁人员信息)

### Structure of batch operation info of face recognition module

|  |
| --- |
| typedef struct tagNETDEVFaceBatchInfo  {  UINT32 udwReqSeq;  UINT32 udwResultCode;  UINT32 udwID;  BYTE byRes[128];  }NETDEV\_FACE\_BATCH\_INFO\_S,\*LPNETDEV\_FACE\_BATCH\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwReqSeq | Sequence number of request data |
| udwResultCode | Returned error code |
| udwID | ID |
| byRes | Reserved field. |

### Structure of entry/exit record info

|  |
| --- |
| typedef struct tagNETDEVACSAttendanceLogInfo  {  UINT32 udwAlarmType;  INT64 tTimeStamp;  CHAR szDoorName[NETDEV\_LEN\_260];  CHAR szDoorNo[NETDEV\_LEN\_64];  UINT32 udwDoorDirect;  CHAR szCardNo[NETDEV\_LEN\_64];  CHAR szPersonName[NETDEV\_LEN\_260];  UINT32 udwPersonType;  CHAR szPersonPhone[NETDEV\_LEN\_64];  CHAR szPersonDept[NETDEV\_LEN\_260];  [NETDEV\_COMPARE\_INFO\_S](#_人脸对比信息结构体) stCompareInfo;  [NETDEV\_FACE\_ATTR\_S](#_人脸属性信息) stFaceAttr;  UINT32 udwCardStatus;  [NETDEV\_CTRL\_LIB\_MATCH\_INFO\_S](#_库比对信息结构体) stLibMatchInfo;  BYTE byRes[256];  }NETDEV\_ACS\_ATTENDANCE\_LOG\_INFO\_S, \*LPNETDEV\_ACS\_ATTENDANCE\_LOG\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwAlarmType | Alarm type |
| tTimeStamp | Alarm time |
| szDoorName | Door name |
| szDoorNo | Door No. |
| udwDoorDirect | In/out direction. 0: In 1: Out |
| szCardNo | Card number |
| szPersonName | Person name |
| udwPersonType | Person type. See [NETDEV\_ACS\_PERSON\_TYPE\_E](#_人员类型枚举). |
| szPersonPhone | Phone number of person |
| szPersonDept | Department of person |
| stCompareInfo | Face comparison info. Required for face recognition terminals. |
| stFaceAttr | Face attribute info |
| udwCardStatus | Card status. 0: Unregistered 1: Registered |
| stLibMatchInfo | Library comparison info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_FindNextACSAttendanceLog](#_获取下一条出入记录)

### Structure of face comparison info

|  |
| --- |
| typedef struct tagNETDEVCompareInfo  {  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stPersonImage;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stSnapshotImage;  BYTE byRes[128];  }NETDEV\_COMPARE\_INFO\_S, \*LPNETDEV\_COMPARE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stPersonImage | Image of person. |
| stSnapshotImage | Snapshot image |
| byRes | Reserved field. |

### Structure of search criteria

|  |
| --- |
| typedef struct tagNETDEVPagedQueryInfo  {  UINT32 udwLimit;  UINT32 udwOffset;  BYTE byRes[128];  }NETDEV\_PAGED\_QUERY\_INFO\_S,\*LPNETDEV\_PAGED\_QUERY\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwLimit | Number of results returned each time. |
| udwOffset | Search starts from the current sequence number |
| byRes | Reserved field |

**See also:**

[[NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表)](#_获取访客黑名单列表)

### Structure of basic info about access control persons

|  |
| --- |
| typedef struct tagNETDEVACSPersonBaseInfo  {  UINT32 udwPersonID;  CHAR szName[NETDEV\_LEN\_260];  UINT32 udwGender;  [NETDEV\_FACE\_MEMBER\_ID\_INFO\_S](#_成员证件信息结构体_1) stMemberIDInfo;  CHAR szTelephone[NETDEV\_LEN\_64];  UINT32 udwCardID;  CHAR szCardNo[NETDEV\_LEN\_16];  UINT32 udwType;  [NETDEV\_ACS\_STAFF\_INFO\_S](#_员工信息结构体_1) stStaffInfo;  [NETDEV\_ACS\_VISITOR\_INFO\_S](#_访客信息结构体_1) stVisitor;  BYTE byRes[256];  }NETDEV\_ACS\_PERSON\_BASE\_INFO\_S, \*LPNETDEV\_ACS\_PERSON\_BASE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPersonID | Person ID |
| szName | Name |
| udwGender | Gender. See [NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举). |
| stMemberIDInfo | ID info |
| szTelephone | Phone number |
| udwCardID | Card ID |
| szCardNo | Card number |
| udwType | Person type. 0: Staff 1: Visitor |
| stStaffInfo | Staff info |
| stVisitor | Visitor info |
| byRes | Reserved field. |

**See also:**

[NETDEV\_FindNextACSPersonInfo](#_获取下一个门禁人员信息)

### Structure of visitor record info

|  |
| --- |
| typedef struct tagACSVisitLogInfo  {  UINT32 udwLogID;  UINT32 udwVisitorID;  CHAR szVisitorName[NETDEV\_LEN\_260];  CHAR szVisitorCompany[NETDEV\_LEN\_260];  CHAR szVisitorPhone[NETDEV\_LEN\_64];  CHAR szCardNo[NETDEV\_LEN\_16];  UINT32 udwIntervieweeID;  CHAR szIntervieweeName[NETDEV\_LEN\_260];  CHAR szIntervieweeDept[NETDEV\_LEN\_260];  INT64 tScheduleStartTime;  INT64 tRealStartTime;  UINT32 udwStatus;  BYTE byRes[256];  }NETDEV\_ACS\_VISIT\_LOG\_INFO\_S, \*LPNETDEV\_ACS\_VISIT\_LOG\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwLogID | Day ID |
| udwVisitorID | Visitor ID |
| szVisitorName | Visitor name. Range: [1,64] characters |
| szVisitorCompany | Company of the visitor. Range: [1,64] characters |
| szVisitorPhone | Visitor's phone number |
| szCardNo | Visitor’s card number |
| udwIntervieweeID | ID of the visited |
| szIntervieweeName | Name of the visited. Range: [1,64] characters |
| szIntervieweeDept | Department of the visited. Range: [1,64] characters |
| tScheduleStartTime | Scheduled visit time (unit: second, UTC format) |
| tRealStartTime | Actual visit time (unit: second, UTC format). |
| udwStatus | Status. See [NETDEV\_ACS\_VISIT\_STATUS\_E](#_访客状态枚举). |
| byRes | Reserved field. |

**See also:**

[[NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录)](#_获取下一条访客记录)

### Structure of door permission info

|  |
| --- |
| typedef struct tagNETDEVACSDoorPermissionInfo  {  UINT32 udwPermissionNum;  [NETDEV\_ACS\_TIME\_SECTION\_S](#_时间信息结构体) stValidTime;  [LPNETDEV\_ACS\_PERMISSION\_GROUP\_INFO\_S](#_权限组信息结构体) pstPermissionGroupList;  BYTE byRes[256];  }NETDEV\_ACS\_DOOR\_PERMISSION\_INFO\_S, \*LPNETDEV\_ACS\_DOOR\_PERMISSION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPermissionNum | Number of permission groups |
| stValidTime | Validity period |
| pstPermissionGroupList | Permission group list. Optional when Num is 0. |
| byRes | Reserved field. |

**See also:**

[NETDEV\_GetACSPersonPermission](#_获取指定人员授权信息), [NETDEV\_SetACSPersonPermission](#_设置指定人员授权信息)

### Structure of permission group info

|  |
| --- |
| typedef struct tagNETDEVACSPermissionGroupInfo  {  UINT32 udwPermissionGroupID ;  CHAR szPermissionGroupName[NETDEV\_LEN\_260];  BYTE byRes[256];  }NETDEV\_ACS\_PERMISSION\_GROUP\_INFO\_S, \*LPNETDEV\_ACS\_PERMISSION\_GROUP\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwPermissionGroupID | Permission group ID |
| szPermissionGroupName | Permission group name |
| byRes | Reserved field. |

### Structure of person verification

|  |
| --- |
| typedef struct tagNETDEVPersonVerification  {  CHAR szReference[NETDEV\_LEN\_128];  UINT32 udwSeq;  UINT32 udwChannelID;  CHAR szChannelName[NETDEV\_LEN\_256];  UINT32 udwTimestamp;  UINT32 udwNotificationType;  UINT32 udwFaceInfoNum;  [LPNETDEV\_CTRL\_FACE\_INFO\_S](#_人脸信息结构体) pstCtrlFaceInfo;  UINT32 udwCardInfoNum;  [LPNETDEV\_CTRL\_CARD\_INFO\_S](#_卡信息结构体) pstCtrlCardInfo;  UINT32 udwGateInfoNum;  [LPNETDEV\_CTRL\_GATE\_INFO\_S](#_闸机信息结构体) pstCtrlGateInfo;  UINT32 udwLibMatInfoNum;  [LPNETDEV\_CTRL\_LIB\_MATCH\_INFO\_S](#_库比对信息结构体) pstLibMatchInfo;  UINT32 udwTemperatureInfoNum;  [LPNETDEV\_CTRL\_TEMPERATURE\_INFO](#_温度信息结构体) pstTemperatureInfo;  BYTE byRes[120];  }NETDEV\_PERSON\_VERIFICATION\_S,\*LPNETDEV\_PERSON\_VERIFICATION\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szReference | Used by the client to determine the URL of the push message |
| udwSeq | Sequence number of notification record |
| udwChannelID | Channel ID. Applicable to VMS. |
| szChannelName | Channel name. Length: [1,64]. Applicable to VMS. |
| udwTimestamp | Notification time (unit: second, UTC format). |
| udwNotificationType | Notification type. 0: real-time notification 1: historical notification |
| udwFaceInfoNum | Number of face info. Range: [0, 1] |
| pstCtrlFaceInfo | Face info. Memory needs to be allocated dynamically. |
| udwCardInfoNum | Number of card info. Range: [0, 1] |
| pstCtrlCardInfo | Card info. Memory needs to be allocated dynamically. |
| udwGateInfoNum | Number of gate info. Range: [0, 1] |
| pstCtrlGateInfo | Gate info. Memory needs to be allocated dynamically. |
| udwLibMatInfoNum | Number of library info compared. Range: [0, 16] |
| pstLibMatchInfo | Library comparison info. Memory needs to be allocated dynamically. |
| udwTemperatureInfoNum | Number of temperature info |
| pstTemperatureInfo | Temperature info list. Memory needs to be allocated dynamically according to udwTemperatureInfoNum. |
| byRes | Reserved field |

**See also:**

[NETDEV\_AlarmMessFGCallBack\_PF](#_注册速通门报警消息回调函数)

### Structure of face info

|  |
| --- |
| typedef struct tagNETDEVCtrlFaceInfo  {  UINT32 udwID;  UINT32 udwTimestamp;  UINT32 udwCapSrc;  UINT32 udwFeatureNum;  [LPNETDEV\_FEATURE\_INFO\_S](#_半结构化特征信息) pstFeatureInfo;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stPanoImage;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stFaceImage;  [NETDEV\_FACE\_POSITION\_INFO\_S](#_人脸位置信息结构体) stFaceArea;  FLOAT fTemperature;  UINT32 udwMaskFlag;  BYTE byRes[120];  }NETDEV\_CTRL\_FACE\_INFO\_S,\*LPNETDEV\_CTRL\_FACE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Record ID |
| udwTimestamp | Collection time (unit: second, UTC format). |
| udwCapSrc | Collection source. See [NETDEV\_CAP\_SRC\_E](#_采集来源枚举). For FaceInfo choose 1. |
| udwFeatureNum | Number of semi-structured attributes. Range: [0, 2]. |
| pstFeatureInfo | List of semi-structured attributes. Memory needs to be allocated dynamically. |
| stPanoImage | Panoramic image |
| stFaceImage | Face image |
| stFaceArea | Coordinates of face area in panoramic image |
| fTemperature | Body temperature (°C, 1 decimal place) |
| udwMaskFlag | Mask 0: Unknown status or detection is disabled 1: no mask 2: wearing a mask |
| byRes | Reserved field |

### Structure of card info

|  |
| --- |
| typedef struct tafNETDEVCtrlCardInfo  {  UINT32 udwID;  UINT32 udwTimestamp;  UINT32 udwCapSrc;  UINT32 udwCardType;  CHAR szCardID[NETDEV\_LEN\_32];  UINT32 udwCardStatus;  CHAR szName[NETDEV\_LEN\_256];  UINT32 udwGender;  CHAR szBirthday[NETDEV\_LEN\_16];  CHAR szResidentialAddress[NETDEV\_LEN\_128];  CHAR szIdentityNo[NETDEV\_LEN\_32];  CHAR szIssuingAuthority[NETDEV\_LEN\_128];  CHAR szIssuingDate[NETDEV\_LEN\_16];  CHAR szValidDateStart[NETDEV\_LEN\_16];  CHAR szValidDateEnd[NETDEV\_LEN\_16];  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stIDImage;  BYTE byRes[132];  }NETDEV\_CTRL\_CARD\_INFO\_S,\*LPNETDEV\_CTRL\_CARD\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Record ID |
| udwTimestamp | Collection time (unit: second, UTC format). |
| udwCapSrc | Collection source. See [NETDEV\_CAP\_SRC\_E](#_采集来源枚举). For CardInfo choose 2 or 3. |
| udwCardType | 0: ID card 1: Access control card |
| szCardID | Field of access control card: Physical card number. Max 18 digits. |
| udwCardStatus | Field of access control card: Card status |
| szName | Field of ID card: Name. Range: [1,63]. |
| udwGender | Field of ID card: Gender. See [NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举). |
| szBirthday | Field of ID card: Birth date (YYYYMMDD). |
| szResidentialAddress | Field of ID card: Address. |
| szIdentityNo | Field of ID card: ID card number. |
| szIssuingAuthority | Field of ID card: Issuing authority |
| szIssuingDate | Field of ID card: Issue date (YYYYMMDD). |
| szValidDateStart | ID card field: Start time of the validity period of the ID card. |
| szValidDateEnd | ID card field: End time of the validity period of the ID card. |
| stIDImage | ID card photo |
| byRes | Reserved field |

### Structure of gate info

|  |
| --- |
| typedef struct tagNETDEVCtrlGateInfo  {  UINT32 udwID;  UINT32 udwTimestamp;  UINT32 udwCapSrc;  UINT32 udwInPersonCnt;  UINT32 udwOutPersonCnt;  BYTE byRes[128];  }NETDEV\_CTRL\_GATE\_INFO\_S,\*LPNETDEV\_CTRL\_GATE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Record ID |
| udwTimestamp | Collection time |
| udwCapSrc | Collection source. See [NETDEV\_CAP\_SRC\_E](#_采集来源枚举). For GateInfo choose 4. |
| udwInPersonCnt | Number of people that have entered |
| udwOutPersonCnt | Number of people that have left |
| byRes | Reserved field |

### Structure of library comparison info

|  |
| --- |
| typedef struct tagNETDEVCtrlLibMatchInfo  {  UINT32 udwID;  UINT32 udwLibID;  UINT32 udwLibType;  UINT32 udwMatchStatus;  UINT32 udwMatchPersonID;  UINT32 udwMatchFaceID;  [NETDEV\_MATCH\_PERSON\_INFO\_S](#_匹配人员信息结构体) stMatchPersonInfo;  BYTE byRes[128];  }NETDEV\_CTRL\_LIB\_MATCH\_INFO\_S,\*LPNETDEV\_CTRL\_LIB\_MATCH\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Record ID |
| udwLibID | Library ID |
| udwLibType | Library type |
| udwMatchStatus | Matching status. See [NETDEV\_MATCH\_STATUS\_E](#_匹配状态枚举). |
| udwMatchPersonID | Matching person ID |
| udwMatchFaceID | Matching face ID |
| stMatchPersonInfo | Matching person info |
| byRes | Reserved field |

### Structure of matching person info

|  |
| --- |
| typedef struct tagNETDEVMatchPersonInfo  {  CHAR szPersonName[NETDEV\_LEN\_256];  UINT32 udwGender;  CHAR szCardID[NETDEV\_LEN\_32];  CHAR szIdentityNo[NETDEV\_LEN\_32];  CHAR szPersonCode[NETDEV\_LEN\_64];  BYTE byRes[64];  }NETDEV\_MATCH\_PERSON\_INFO\_S,\*LPNETDEV\_MATCH\_PERSON\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szPersonName | Member name. Range: [1,63]. |
| udwGender | Member gender. See [NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举). |
| szCardID | Access control card number |
| szIdentityNo | ID card number |
| szPersonCode | Person ID. Can be filled with student ID or employee ID. Range: [1, 15] |
| byRes | Reserved field |

### Structure of image info

|  |
| --- |
| typedef struct tagNETDEVStructImageInfo  {  UINT32 udwIndex;  UINT32 udwType;  UINT32 udwFormat;  UINT32 udwWidth;  UINT32 udwHeight;  UINT32 udwCaptureTime;  CHAR szUrl[NETDEV\_LEN\_260];  UINT32 udwSize;  CHAR\* pszData;  BYTE byRes[128];  }NETDEV\_STRUCT\_IMAGE\_INFO\_S, \*LPNETDEV\_STRUCT\_IMAGE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwIndex | Image index |
| udwType | Image type |
| udwFormat | Image format. See [NETDEV\_IMAGE\_FORMAT\_E](#_图像格式枚举) |
| udwWidth | Image width |
| udwHeight | Image height |
| udwCaptureTime | Image capture time |
| szUrl | Image URL |
| udwSize | Size of Base64-encoded image. Max 3MB. |
| pszData | Base64-encoded image data |
| byRes | Reserved field. |

### Structure of plate monitoring and alarm info

|  |
| --- |
| typedef struct tagstNETDEVMonitorAlarmInfo  {  UINT32 udwMonitorReason;  UINT32 udwMonitorAlarmType;  UINT32 udwMemberID;  }NETDEV\_MONITOR\_ALARM\_INFO\_S,\*LPNETDEV\_MONITOR\_ALARM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwMonitorReason | Monitoring cause |
| udwMonitorAlarmType | Monitoring alarm type 0: Match alarm 1: Not match alarm |
| udwMemberID | Vehicle member ID |

### Vehicle member info

|  |
| --- |
| typedef struct tagstNETDEVVehicleDetailInfo  {  UINT32 udwReqSeq;  UINT32 udwMemberID;  [NETDEV\_PLATE\_ATTR\_INFO\_S](#_车牌信息) stPlateAttr;  [NETDEV\_VEHICLE\_MEMBER\_ATTR\_S](#_车辆信息_1) stVehicleAttr;  BOOL bIsMonitored;  UINT32 udwDBNum;  UINT32 audwDBIDList[NETDEV\_LEN\_16];  BYTE byRes[124];  }NETDEV\_VEHICLE\_DETAIL\_INFO\_S, \*LPNETDEV\_VEHICLE\_DETAIL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwReqSeq | Sequence number of request data |
| udwMemberID | Face member ID |
| stPlateAttr | Plate info |
| stVehicleAttr | Vehicle info |
| bIsMonitored | Whether monitoring is configured 0: No 1: Yes |
| udwDBNum | Number of vehicle libraries containing the member. |
| audwDBIDList | Array of IDs of the vehicle libraries containing the member. |
| byRes | Reserved field. |

### Vehicle info

|  |
| --- |
| typedef struct tagNETDEVVehicleMemberAttr  {  UINT32 udwColor;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stVehicleImage;  }NETDEV\_VEHICLE\_MEMBER\_ATTR\_S,\*LPNETDEV\_VEHICLE\_MEMBER\_ATTR\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwColor | Vehicle color. See [NETDEV\_PLATE\_COLOR\_E](#_车牌颜色枚举). |
| stVehicleImage | Vehicle image. Max 4MB after encryption |

### Plate info

|  |
| --- |
| typedef struct tagstNETDEVPlateAttrInfo  {  CHAR szPlateNo[NETDEV\_LEN\_16];  UINT32 udwColor;  UINT32 udwType;  BYTE byRes[64];  }NETDEV\_PLATE\_ATTR\_INFO\_S, \*LPNETDEV\_PLATE\_ATTR\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szPlateNo | Plate No. |
| udwColor | Plate color. See [NETDEV\_PLATE\_COLOR\_E](#_车牌颜色枚举). |
| udwType | Plate type. See [NETDEV\_PLATE\_TYPE\_E](#_车牌类型). |
| byRes | Reserved field. |

### Structure of vehicle list

|  |
| --- |
| typedef struct tagNETDEVVehicleInfoList  {  UINT32 udwVehicleNum;  [LPNETDEV\_VEHICLE\_DETAIL\_INFO\_S](#_车辆成员信息) pstMemberInfoList;  BYTE byRes[132];  }NETDEV\_VEHICLE\_INFO\_LIST\_S,\*LPNETDEV\_VEHICLE\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwVehicleNum | Number of vehicle members |
| pstMemberInfoList | Vehicle member list. Memory needs to be allocated dynamically according to udwNum. |
| byRes | Reserved field. |

### Structure of batch operation member list

|  |
| --- |
| typedef struct tagNETDEVBatchOperateMemberList  {  UINT32 udwTaskNo;  UINT32 udwMemberNum;  UINT32\* pstMemberIDList;  BYTE byRes[128];  }NETDEV\_BATCH\_OPERATE\_MEMBER\_LIST\_S,\*LPNETDEV\_BATCH\_OPERATE\_MEMBER\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTaskNo | Operation task ID. Applicable to NVR only. |
| udwMemberNum | Number of members |
| pstMemberIDList | Member list. Memory needs to be allocated dynamically according to udwNum. |
| byRes | Reserved field. |

### Structure of returned alarm records

|  |
| --- |
| typedef struct tagstNETDEVSmartAlarmLogResultInfo  {  UINT32 udwTotal;  UINT32 udwOffset;  UINT32 udwNum;  BYTE byRes[128];  }NETDEV\_SMART\_ALARM\_LOG\_RESULT\_INFO\_S, \*LPNETDEV\_SMART\_ALARM\_LOG\_RESULT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTotal | Total number of alarm records |
| udwOffset | Record offset |
| udwNum | Number of alarm records returns this time |
| byRes | Reserved field. |

### Structure of vehicle attribute info

|  |
| --- |
| typedef struct tagNETDEVVehAttr  {  UINT32 udwType;  UINT32 udwColor;  UINT32 udwSpeedUnit;  FLOAT fSpeedValue;  UINT32 udwSpeedType;  CHAR szVehicleBrand[NETDEV\_LEN\_64];  UINT32 udwImageDirection;  BYTE byRes[128];  }NETDEV\_VEH\_ATTR\_S,\*LPNETDEV\_VEH\_ATTR\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | Vehicle type. See NETDEV\_VEHICLE\_TYPE\_E. |
| udwColor | Vehicle color. See NETDEV\_PLATE\_COLOR\_E. |
| udwSpeedUnit | Vehicle speed unit. 0: kph 1: mph |
| fSpeedValue | Vehicle speed |
| udwSpeedType | Vehicle speed type in structured scenes. See NETDEV\_SPEED\_TYPE\_E. |
| szVehicleBrand | Vehicle brand (custom) |
| udwImageDirection | Driving direction (relative to the image) of motor vehicle in structured scenes. See NETDEV\_IMAGE\_DIRECTION\_E. |
| byRes | Reserved field. |

### Structure of plate monitoring and alarm info

|  |
| --- |
| typedef struct tagstNETDEVMonitorAlarmInfo  {  UINT32 udwMonitorReason;  UINT32 udwMonitorAlarmType;  UINT32 udwMemberID;  }NETDEV\_MONITOR\_ALARM\_INFO\_S,\*LPNETDEV\_MONITOR\_ALARM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwMonitorReason | Type of monitoring cause |
| udwMonitorAlarmType | Monitoring alarm type 0: Match alarm 1: Not match alarm |
| udwMemberID | Vehicle member ID |

**See also:**

### Structure of transparent channel creation

|  |
| --- |
| typedef struct tagNETDEVSerialStart  {  INT32 dwSerialPort;  INT32 dwSerialNum;  BYTE byRes[512];  }NETDEV\_SERIAL\_START\_S, \*LPNETDEV\_SERIAL\_START\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSerialPort | Serial port type. 1: RS232 2: RS454. Currently only RS485 is supported. |
| dwSerialNum | Serial port ID |
| byRes | Reserved field. |

### Structure of user info

|  |
| --- |
| typedef struct tagNETDEVUserDetailInfo  {  UINT32 udwLevel;  CHAR szUserName[NETDEV\_LEN\_64];  CHAR szPassword[NETDEV\_LEN\_256];  UINT32 udwBasePermission;  UINT32 udwNum;  [NETDEV\_CHN\_PERMISSION\_INFO\_S](#_通道权限信息结构体) astChnPermission[NETDEV\_CHANNEL\_MAX];  BYTE byRes[256];  }NETDEV\_USER\_DETAIL\_INFO\_S, \*LPNETDEV\_USER\_DETAIL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwLevel | User level. See NETDEV\_USER\_LEVEL\_E. |
| szUserName | Username. Unmodifiable. Used as input parameter to get info of a user. Range: 1-18. |
| szPassword | Password. Used when adding a user. Range: 0-256. Will not be returned when obtaining user info. |
| udwBasePermission | Basic permission. See [NETDEV\_USER\_BASE\_PERMISSION\_E](#_基本权限枚举). Described by BIT. 1: permission is effective. 0: permission is not effective. |
| udwNum | Number of video input channels |
| astChnPermission | Channel permission list |
| byRes | Reserved field. |

### Structure of channel permission info

|  |
| --- |
| typedef struct tagNETDEVChnPermissionInfo  {  INT32 dwChannelID;  INT32 dwPermission;  BYTE byRes[256];  }NETDEV\_CHN\_PERMISSION\_INFO\_S, \*LPNETDEV\_CHN\_PERMISSION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwChannelID | Channel ID |
| dwPermission | Channel permission. See [NETDEV\_CHN\_PERMISSION\_TYPE\_E](#_通道权限枚举). |
| byRes | Reserved field. |

### Structure of modifying detailed user info

|  |
| --- |
| typedef struct tagNETDEVUserModifyDetailInfo  {  [NETDEV\_USER\_DETAIL\_INFO\_S](#_用户信息结构体) stUserInfo;  BOOL bIsModifyPassword;  BOOL bIsModifyOther;  CHAR szNewPassword[NETDEV\_LEN\_256];  CHAR szCurrentPassword[NETDEV\_LEN\_256];  BYTE byRes[256];  }NETDEV\_USER\_MODIFY\_DETAIL\_INFO\_S, \*LPNETDEV\_USER\_MODIFY\_DETAIL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stUserInfo | User info |
| bIsModifyPassword | Whether change password. Assign value during modification. |
| bIsModifyOther | Whether it is admin modifying other user's password. Assign value during modification. |
| szNewPassword | New password (required when changing password). Range: 0-256. If the configured password is empty, then the empty field will be passed in. |
| szCurrentPassword | Current password (required when changing password). Range: 0-256. Only the new password is required when admin changes the password of other users. If the configured password is empty, then the empty field will be passed in. |
| byRes | Reserved field. |

### Structure of user password info

|  |
| --- |
| typedef struct tagNETDEVModifyPasswordInfo  {  CHAR szUserName[NETDEV\_LEN\_260];  CHAR szNewPIN[NETDEV\_LEN\_256];  CHAR szCurrentPIN[NETDEV\_LEN\_256];  BYTE byRes[128];  }NETDEV\_MODIFY\_PASSWORD\_INFO\_S,\*LPNETDEV\_MODIFY\_PASSWORD\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szUserName | Username |
| szNewPIN | New password |
| szCurrentPIN | Old password |
| byRes | Reserved field. |

### Structure of user list

|  |
| --- |
| typedef struct tagNETDEVUserDetailList  {  UINT32 udwNum;  [NETDEV\_USER\_DETAIL\_INFO\_S](#_用户信息结构体) astUserInfo[NETDEV\_LEN\_64];  BYTE byRes[128];  }NETDEV\_USER\_DETAIL\_LIST\_S, \*LPNETDEV\_USER\_DETAIL\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of users |
| astUserInfo | User list |
| byRes | Reserved field. |

### Structure of organization info

|  |
| --- |
| typedef struct tagstNETDEVOrgDetailInfo  {  INT32 dwOrgID;  INT32 dwParentID;  INT32 dwType;  CHAR szNodeName[NETDEV\_NAME\_MAX\_LEN];  CHAR szDesc[NETDEV\_DESCRIBE\_MAX\_LEN];  UINT32 udwTime;  CHAR szUserName[NETDEV\_LEN\_64];  BYTE byRes[60];  }NETDEV\_ORG\_INFO\_S, \*LPNETDEV\_ORG\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwOrgID | Organization ID |
| dwParentID | Parent node ID |
| dwType | Type. See [NETDEV\_ORG\_TYPE\_E](#_组织类型枚举). |
| szNodeName | Node name |
| szDesc | Remarks |
| udwTime | Creation time (UTC) |
| szUserName | Creator [1,64] |
| byRes | Reserved field. |

### Structure of info about organizations to be deleted

|  |
| --- |
| typedef struct tagstNETDEVDelOrgDetailInfo  {  INT32 dwOrgNum;  INT32 \*pdwOrgIDs;  INT32 dwOrgType;  BYTE byRes[64];  }NETDEV\_DEL\_ORG\_INFO\_S, \*LPNETDEV\_DEL\_ORG\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwOrgNum | Number of organizations |
| pdwOrgIDs | ID of organization to be deleted. Memory needs to be allocated dynamically according to dwOrgNum. |
| dwOrgType | Organization type. See [NETDEV\_ORG\_TYPE\_E](#_组织类型枚举). |
| byRes | Reserved field. |

### Structure of response to organization deletion

|  |
| --- |
| typedef struct tagstNETDEVOrgDelDetailInfo  {  INT32 dwStatus;  INT32 dwNum;  [LPNETDEV\_OPERATE\_INFO\_S](#_通道操作信息结构体) pstResultInfo;  BYTE byRes[68];  }NETDEV\_ORG\_BATCH\_DEL\_INFO\_S, \*LPNETDEV\_ORG\_BATCH\_DEL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwStatus | Response status. See [NETDEV\_ORG\_RESPONSE\_STAUTE\_E](#_响应状态类型枚举). |
| dwNum | Number of responses |
| pstResultInfo | Returned batch operation results. Memory needs to be allocated dynamically according to the number of organizations to be deleted. |
| byRes | Reserved field. |

### Structure of info about channels in an organization

|  |
| --- |
| typedef struct tagstNETDEVOrgChnShortInfo  {  INT32 dwOrgID;  INT32 dwChannelsNum;  INT32 \*pdwChnIDs;  BYTE byRes[68];  }NETDEV\_ORG\_CHN\_SHORT\_INFO\_S, \*LPNETDEV\_ORG\_CHN\_SHORT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwOrgID | Organization ID |
| dwChannelsNum | Number of channels |
| pdwChnIDs | Channel ID. Pointer. RAM is allocated dynamically according to dwChannelsNum. |
| byRes | Reserved field. |

### Structure of info about channels in an organization

|  |
| --- |
| typedef struct tagstNETDEVOrgChnInfo  {  INT32 dwOrgID;  INT32 dwChannelID;  CHAR szChnName[NETDEV\_NAME\_MAX\_LEN];  INT32 dwChnIndex;  INT32 dwChnType;  INT32 dwChnStatus;  INT32 dwDevID;  CHAR szDevName[NETDEV\_NAME\_MAX\_LEN];  BYTE byRes[128];  }NETDEV\_ORG\_CHN\_INFO\_S, \*LPNETDEV\_ORG\_CHN\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwOrgID | Organization ID |
| dwChannelID | Channel ID |
| szChnName | Channel name |
| dwChnIndex | Channel index |
| dwChnType | Channel type |
| dwChnStatus | Channel status |
| dwDevID | Device ID |
| szDevName | Device name |
| byRes | Reserved field. |

### Structure of info about network port number and status

|  |
| --- |
| typedef struct tagNETDEVUpnpNatState  {  INT32 dwSize;  [NETDEV\_UPNP\_PORT\_STATE\_S](#_协议信息结构体) astUpnpPort[NETDEV\_LEN\_16];  }NETDEV\_UPNP\_NAT\_STATE\_S, \*LPNETDEV\_UPNP\_NAT\_STATE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwSize | Number of protocols |
| astUpnpPort | Protocol info |

### Structure of protocol info

|  |
| --- |
| typedef struct tagNETDEVUpnpPortState  {  NETDEV\_PROTOCOL\_TYPE\_E eType;  BOOL bEnbale;  INT32 dwPort;  BYTE byRes[128];  }NETDEV\_UPNP\_PORT\_STATE\_S, \*LPNETDEV\_UPNP\_PORT\_STATE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| eType | Protocol type. See [NETDEV\_PROTOCOL\_TYPE\_E](#_协议类型枚举). |
| bEnbale | Whether is supported |
| dwPort | Port number |
| byRes | Reserved field. |

### Structure of monitoring info search criteria

|  |
| --- |
| typedef struct tagNETDEVMonitorQueryInfo  {  UINT32 udwLimit;  UINT32 udwOffset;  BOOL bIsQueryAll;  BYTE byRes[128];  }NETDEV\_MONITOR\_QUERY\_INFO\_S, \*LPNETDEV\_MONITOR\_QUERY\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwLimit | Search limit. Max: 20. |
| udwOffset | Sequence number from where the search starts. Starts from 0. |
| bIsQueryAll | Whether to search all. Yes: TRUE No: FALSE |
| byRes | Reserved field. |

### Structure of criteria for searching organization list

|  |
| --- |
| typedef struct tagstNETDEVOrgFindCond  {  UINT32 udwOrgType;  UINT32 udwRootOrgID;  UINT32 udwFindType;  BYTE byRes[128];  }NETDEV\_ORG\_FIND\_COND\_S, \*LPNETDEV\_ORG\_FIND\_COND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwOrgType | Organization type. See [NETDEV\_ORG\_TYPE\_E](#_组织类型枚举). |
| udwRootOrgID | Root organization ID |
| udwFindType | Search mode. See [NETDEV\_ORG\_FIND\_MODE\_E](#_组织查找模式枚举). |
| byRes | Reserved field. |

### Area people counting rule info

|  |
| --- |
| typedef struct tagNETDEVAreaPeopleCountRuleInfo  {  BOOL bEnabled;  UINT32 udwReportInterval;  UINT32 udwAreaNum;  LPNETDEV\_DETECT\_AREA\_RULE\_INFO\_S pstDetectAreaRuleInfo;  BYTE byRes[512];  }NETDEV\_AREA\_PEOPLE\_COUNT\_RULE\_INFO\_S,\*LPNETDEV\_AREA\_PEOPLE\_COUNT\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Whether is enabled. FALSE: No TRUE: Yes |
| udwReportInterval | Report time interval (unit: second). Value range: [1, 60] |
| udwAreaNum | Number of detection areas |
| pstDetectAreaRuleInfo | Info about detection area rules. Memory needs to be allocated dynamically according to udwAreaNum. See [NETDEV\_DETECT\_AREA\_RULE\_INFO\_S](#_检测区域规则信息). |
| byRes | Reserved field. |

### Area detection rule info

|  |
| --- |
| typedef struct tagNETDEVDetectAreaRuleInfo  {  UINT32 udwDetectAreaID;  BOOL bEnabled;  NETDEV\_AREA\_LOCATION\_INFO\_S stAreaLocationInfo;  BYTE byRes[512];  }NETDEV\_DETECT\_AREA\_RULE\_INFO\_S,\*LPNETDEV\_DETECT\_AREA\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwDetectAreaID | Detection area ID. Starts from 0. |
| bEnabled | Whether is enabled. FALSE: No TRUE: Yes |
| udwAreaNum | Number of detection areas |
| stAreaLocationInfo | Area location info. See [NETDEV\_AREA\_LOCATION\_INFO\_S](#_区域位置信息). |
| byRes | Reserved field. |

### Area location info

|  |
| --- |
| typedef struct tagNETDEVAreaLocationInfo  {  UINT32 udwAreaPointNum;  LPNETDEV\_AREA\_POINT\_COORDINATE\_S pstAreaPointCoordinate;  BYTE byRes[512];  }NETDEV\_AREA\_LOCATION\_INFO\_S,\*LPNETDEV\_AREA\_LOCATION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwAreaPointNum | Number of vertices in detection area. Range: [3,6] |
| pstAreaPointCoordinate | Coordinates of each vertex of the detection area. The origin of the coordinates is in the upper left corner. See [NETDEV\_AREA\_POINT\_COORDINATE\_S](#_绘制规则区域中每个顶点的坐标). |
| byRes | Reserved field. |

### Coordinates of each vertex of the detection area

|  |
| --- |
| typedef struct tagNETDEVAreaPointCoordinate  {  UINT32 udwX;  UINT32 udwY;  }NETDEV\_AREA\_POINT\_COORDINATE\_S,\*LPNETDEV\_AREA\_POINT\_COORDINATE\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwX | x-coordinate (‱). Range: [0,9999] |
| udwY | y-coordinate (‱). Range: [0,9999] |

### Tripwire people counting rule info

|  |
| --- |
| typedef struct tagNETDEVLinePeopleCountRuleInfo  {  BOOL bEnabled;  UINT32 udwReportInterval;  UINT32 udwLineNum;  LPNETDEV\_DETECT\_LINE\_RULE\_INFO\_S pstDetectLineRuleInfo;  BYTE byRes[512];  }NETDEV\_LINE\_PEOPLE\_COUNT\_RULE\_INFO\_S,\*LPNETDEV\_LINE\_PEOPLE\_COUNT\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Whether is enabled. FALSE: No TRUE: Yes |
| udwReportInterval | Report time interval (unit: second). Value range: [1, 60] |
| udwLineNum | Number of tripwires |
| pstDetectLineRuleInfo | Info about detection tripwire rules. Memory needs to be allocated dynamically according to udwLineNum. See [NETDEV\_DETECT\_LINE\_RULE\_INFO\_S](#_检测绊线规则信息). |
| byRes | Reserved field. |

### Tripwire detection rule info

|  |
| --- |
| typedef struct tagNETDEVDetectLineRuleInfo  {  UINT32 udwDetectLineID;  BOOL bEnabled;  UINT32 udwInDirection;  UINT32 udwStatisticType;  NETDEV\_AREA\_POINT\_COORDINATE\_S stStartPointCoordinate;  NETDEV\_AREA\_POINT\_COORDINATE\_S stEndPointCoordinate;  BYTE byRes[512];  }NETDEV\_DETECT\_LINE\_RULE\_INFO\_S,\*LPNETDEV\_DETECT\_LINE\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwDetectLineID | Detection area ID. Starts from 0. |
| bEnabled | Whether is enabled. FALSE: No TRUE: Yes |
| udwInDirection | Entry direction 1：B->A, from position B to position A；2:A->B, from position A to position B |
| udwStatisticType | Counting type. See [NETDEV\_STATISTIC\_TYPE\_E](#_人数统计类型). |
| stStartPointCoordinate | Coordinates of the start point of the detection line. See [NETDEV\_AREA\_POINT\_COORDINATE\_S](#_绘制规则区域中每个顶点的坐标). |
| stEndPointCoordinate | Coordinates of the end point of the detection line. See [NETDEV\_AREA\_POINT\_COORDINATE\_S](#_绘制规则区域中每个顶点的坐标). |
| byRes | Reserved field. |

### Tripwire people counting reset info

|  |
| --- |
| typedef struct tagNETDEVLinePeopleCountResetInfo  {  BOOL bTimingResetEnabled;  CHAR szResetTime[NETDEV\_LEN\_64];  BYTE byRes[256];  }NETDEV\_LINE\_PEOPLE\_COUNT\_RESET\_INFO\_S,\*LPNETDEV\_LINE\_PEOPLE\_COUNT\_RESET\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bTimingResetEnabled | Enable or disable timed reset. False: Disable TRUE: Enable |
| szResetTime | Time when counting will be reset. String length: [0,24]. Format: 00:00:00 Required when timed reset is enabled |
| byRes | Reserved field. |

### List of linked dome cameras

|  |
| --- |
| typedef struct tagNETDEVLinkDomeInfoList  {  UINT32 udwLinkDomeNum;  NETDEV\_LINK\_DOME\_INFO\_S stLinkDomeInfoList[NETDEV\_LEN\_8];  BYTE byRes[128];  }NETDEV\_LINK\_DOME\_INFO\_LIST\_S, \*LPNETDEV\_LINK\_DOME\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwLinkDomeNum | Number of dome cameras |
| stLinkDomeInfoList | Dome camera list. See [NETDEV\_LINK\_DOME\_INFO\_S](#_联动球机信息). |
| byRes | Reserved field. |

### Info about linked dome camera

|  |
| --- |
| typedef struct tagNETDEVLinkDomeInfo  {  UINT32 udwDomeID;  CHAR szIPAddr[NETDEV\_LEN\_64];  CHAR szUserName[NETDEV\_LEN\_64];  CHAR szPIN[NETDEV\_LEN\_64];  UINT32 udwHttpPort;  UINT32 udwZoomCoefficient;  BYTE byRes[128];  }NETDEV\_LINK\_DOME\_INFO\_S, \*LPNETDEV\_LINK\_DOME\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwDomeID | Dome ID. Starts from 0. |
| szIPAddr | IP address |
| szUserName | Username. Length: [0,64] |
| szPIN | Password. Length: [0,64] |
| udwHttpPort | HTTP port |
| udwZoomCoefficient | Zoom coefficient. Range: [1-10] |
| byRes | Reserved field. |

### System time configuration

|  |
| --- |
| typedef struct tagNETDEVSystemTimeInfo  {  [NETDEV\_TIME\_ZONE\_E](#_时区) dwTimeZone;  INT64 tUtcTime;  UINT32 udwDateFormat;  UINT32 udwHourFormat;  BYTE byRes[212];  }NETDEV\_TIME\_CFG\_S, \*LPNETDEV\_TIME\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwTimeZone | Time zone. See [NETDEV\_TIME\_ZONE\_E](#_时区). |
| tUtcTime | UTC time |
| udwDateFormat | Date format. 0: YYYY-MM-DD 1: MM-DD-YYYY 2: DD-MM-YYYY |
| udwHourFormat | Time format. 0: 12H format 1: 24H format |
| byRes | Reserved field. |

### Structure of object list

|  |
| --- |
| typedef struct tagNETDEVObjectList  {  UINT32 udwObjectType;  UINT32 udwObjectID;  BYTE byRes[256];  }NETDEV\_OBJECT\_LIST\_S, \*LPNETDEV\_OBJECT\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwObjectType | Object type. See [NETDEV\_OBJECT\_TYPE\_E](#_目标类型枚举). |
| udwObjectID | Object ID |
| byRes | Reserved field. |

### Structure of VIID info

|  |
| --- |
| typedef struct tagNETDEVViidCfg  {  UINT32 udwCoordinateType;  CHAR szViidServerID[NETDEV\_LEN\_64];  UINT32 udwConnectMode;  UINT32 udwObjectType;  BYTE byRes[32];  }NETDEV\_VIID\_CFG\_S, \*LPNETDEV\_VIID\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwCoordinateType | Coordinates type. 0: ‱, (range: [0,9999]) 1: Pixel coordinates 2: Normalized coordinates (range: 0~1). NETDEV\_INVALID\_PARAM means invalid value. |
| szViidServerID | Server ID. Length: [1,32]. Applicable to IPC. |
| udwConnectMode | Connection mode. 0: Standard mode 1: Non-persistent connection. NETDEV\_INVALID\_PARAM means invalid value. |
| udwObjectType | Object type (only used for VIID, and optional when the specified object type needs to be reported). Combination by bit. Each bit corresponds to an object type. 0 means “not report”, and 1 means “report”. Bit0: Face Bit1: Human body Bit2: Motor vehicle Bit3: Non-motor vehicle |
| byRes | Reserved field. |

### Structure of temperature info

|  |
| --- |
| typedef struct tagNETDEVCtrlTemperatureInfo  {  UINT32 udwRelativeFaceID;  FLOAT fEnvTemperature;  FLOAT fTemperatureThreshold;  FLOAT fBodyTemperature;  BYTE byRes[256];  }NETDEV\_CTRL\_TEMPERATURE\_INFO, \*LPNETDEV\_CTRL\_TEMPERATURE\_INFO; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwRelativeFaceID | Related face ID. If no face is related, fill in 0xffffffff |
| fEnvTemperature | Ambient temperature (°C) |
| fTemperatureThreshold | Temperature threshold (°C) |
| fBodyTemperature | Body temperature (°C) |
| byRes | Reserved field. |

### Recording time info

|  |
| --- |
| typedef struct tagNETDEVRecordTime  {  UINT32 udwChlID;  INT64 tEarliestTime;  INT64 tLatestTime;  BYTE byRes[512];  }NETDEV\_RECORD\_TIME\_S, \*LPNETDEV\_RECORD\_TIME\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwChlID | Channel ID |
| tEarliestTime | Earliest time |
| tLatestTime | Latest time |
| byRes | Reserved field. |

### Recording time list

|  |
| --- |
| typedef struct tagNETDEVRecordTimeList  {  UINT32 udwNum;  LPNETDEV\_RECORD\_TIME\_S pstRecordTimes;  BYTE byRes[512];  }NETDEV\_RECORD\_TIME\_LIST\_S, \*LPNETDEV\_RECORD\_TIME\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of recording times |
| pstRecordTimes | Recording time list. Memory needs to be allocated dynamically. See [NETDEV\_RECORD\_TIME\_S](#_录像时间信息). |
| byRes | Reserved field. |

### Detection area vertex coordinates info

|  |
| --- |
| typedef struct tagNETDEVPointInfo  {  UINT32 udwX;  UINT32 udwY;  BYTE byRes[32];  }NETDEV\_POINT\_INFO\_S,\*LPNETDEV\_POINT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwX | x-coordinate. Range: [0,10000] |
| udwY | y-coordinate. Range: [0,10000] |
| byRes | Reserved field. |

### Rule info

|  |
| --- |
| typedef struct tagNETDEVRuleInfo  {  UINT32 udwRuleType;  UINT32 udwTrigerType;  UINT32 udwPointNum;  [LPNETDEV\_POINT\_INFO\_S](#_检测区域图形定点坐标信息) pstPointInfo;  BYTE byRes[64];  }NETDEV\_RULE\_INFO\_S,\*LPNETDEV\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwRuleType | Rule type. See [NETDEV\_RULE\_TYPE\_E](#_规则类型枚举). |
| udwTrigerType | Rule trigger type. See [NETDEV\_RULE\_TRIGGER\_TYPE\_E](#_规则触发类型枚举). |
| udwPointNum | Number of vertices of detection area |
| pstPointInfo | Coordinates of the vertices of the detection area. Optional when udwPointNum is 0. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Info on a single object

|  |
| --- |
| typedef struct tagNETDEVSingleObjectInfo  {  UINT32 udwObjectType;  [NETDEV\_FACE\_STRUCT\_INFO\_S](#_人脸信息) stFaceInfo;  [NETDEV\_PERSON\_STRUCT\_INFO\_S](#_人员信息) stPersonInfo;  [NETDEV\_NON\_MOTOR\_VEH\_INFO\_S](#_非机动车信息) stNonMotorVehInfo;  [NETDEV\_VEH\_INFO\_S](#_车辆信息) stVehInfo;  BYTE byRes[256];  }NETDEV\_SINGLE\_OBJECT\_INFO\_S,\*LPNETDEV\_SINGLE\_OBJECT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwObjectType | Object type. See [NETDEV\_OBJECT\_TYPE\_E](#_目标类型枚举). |
| stFaceInfo | Face information |
| stPersonInfo | Person information |
| stNonMotorVehInfo | Non-motor vehicle information |
| stVehInfo | Vehicle information |
| byRes | Reserved field. |

### Audio output parameters

|  |
| --- |
| typedef struct tagNETDEVAudioOutputCfg  {  UINT32 udwType;  UINT32 udwGain;  BYTE byRes[256];  }NETDEV\_AUDIO\_OUTPUT\_CFG\_S,\*LPNETDEV\_AUDIO\_OUTPUT\_CFG\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | Type of audio output device. See [NETDEV\_AUDIO\_CHL\_MODE\_E](#_音频输入模式枚举). |
| udwGain | Audio output gain. Range: [0,255]. Optional when AudioOutTypeNum is 0. |
| byRes | Reserved field. |

### Audio file information

|  |
| --- |
| typedef struct tagNETDEVAudioFileInfo  {  UINT32 udwID;  UINT32 udwStatus;  UINT32 udwInstallType;  CHAR szFileName[NETDEV\_LEN\_1024];  BYTE byRes[256];  }NETDEV\_AUDIO\_FILE\_INFO\_S,\*LPNETDEV\_AUDIO\_FILE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwID | Audio file ID |
| udwStatus | Audio file status. 0: Disabled 1: Enabled |
| udwInstallType | Installation type. 0: Built-in 1: External |
| szFileName | Audio file name. Max 256 bytes. |
| byRes | Reserved field. |

### Channel list

|  |
| --- |
| typedef struct tagNETDEVChannelsInfos  {  UINT32 udwNum;  UINT32 audwChannelList[NETDEV\_CHANNEL\_MAX];  BYTE byRes[128];  }NETDEV\_CHANNEL\_LIST\_S, \*LPNETDEV\_CHANNEL\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of connected channels. Must not be 0. |
| audwChannelList | List of IDs of connected channels |
| byRes | Reserved field. |

### Structure of channel search criteria

|  |
| --- |
| typedef struct tagQueryCondition  {  INT32 dwLimit;  INT32 dwOffset;  INT32 dwQryInfoNum;  [LPNETDEV\_QUERY\_INFO\_S](#_告警日志查询条件结构体) pstQueryInfo;  INT32 dwRecursion;  BYTE byRes[260];  }NETDEV\_QUERY\_CHN\_CONDITION\_S, \*LPNETDEV\_QUERY\_CHN\_CONDITION\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| dwLimit | Number of results returned each time. |
| dwOffset | Starts search from the current sequence number |
| dwQryInfoNum | Number of search criteria |
| pstQueryInfo | Search criteria. Memory needs to be allocated dynamically. |
| dwRecursion | Recursive query type. 0: None 1: Upward 2: Downward |
| byRes | Reserved field. |

### Image search criteria

|  |
| --- |
| typedef struct tagNETDEVPicQueryCond  {  UINT32 udwSearchID;  UINT32 udwLimit;  UINT32 udwOffset;  BYTE byRes[128];  }NETDEV\_PIC\_QUERY\_COND\_S,\*LPNETDEV\_PIC\_QUERY\_COND\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwSearchID | Service ID |
| udwLimit | Number of results returned each time. |
| udwOffset | Sequence number from where the search starts. The sequence number starts from 0. |
| byRes | Reserved field. |

### Image search results

|  |
| --- |
| typedef struct tagNETDEVPicQueryResult  {  UINT32 udwTotal;  UINT32 udwNum;  BYTE byRes[128];  }NETDEV\_PIC\_QUERY\_RESULT\_S,\*LPNETDEV\_PIC\_QUERY\_RESULT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTotal | Total number of search results |
| udwNum | Number of results returned |
| byRes | Reserved field. |

### Object result info

|  |
| --- |
| typedef struct tagNETDEVObjectResultInfo  {  UINT32 udwRecordID;  UINT32 udwType;  UINT32 udwTime;  CHAR szChannelName[NETDEV\_LEN\_256];  UINT32 udwChannelID;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stBigImageInfo;  UINT32 udwObjectInfoNum;  [LPNETDEV\_OBJECT\_INFO\_LIST\_S](#_目标信息列表) pstObjectInfoList;  BYTE byRes[512];  }NETDEV\_OBJECT\_RESULT\_INFO\_S,\*LPNETDEV\_OBJECT\_RESULT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwRecordID | Record ID |
| udwType | Record type. See [NETDEV\_OBJECT\_RECORD\_TYPE\_E](#_目标记录类型). |
| udwTime | Record time (unit: second, UTC format). |
| szChannelName | Capture channel name. Range: [1, 63]. |
| udwChannelID | Capture channel ID |
| stBigImageInfo | Info about complete image |
| udwObjectInfoNum | Number of object info |
| pstObjectInfoList | Object list. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Object info list

|  |
| --- |
| typedef struct tagNETDEVObjectInfoList  {  UINT32 udwObjectID;  [NETDEV\_FILE\_INFO\_S](#_文件信息结构体) stSmallImageInfo;  [NETDEV\_SINGLE\_OBJECT\_INFO\_S](#_单个目标信息) stObjectInfo;  BYTE byRes[1024];  }NETDEV\_OBJECT\_INFO\_LIST\_S,\*LPNETDEV\_OBJECT\_INFO\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwRecordID | Record ID |
| udwType | Record type. See [NETDEV\_OBJECT\_RECORD\_TYPE\_E](#_目标记录类型). |
| udwTime | Record time (unit: second, UTC format). |
| szChannelName | Capture channel name. Range: [1, 63]. |
| udwChannelID | Capture channel ID |
| stBigImageInfo | Info about complete image |
| udwObjectInfoNum | Number of object info |
| pstObjectInfoList | Object list. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Alarm related data

|  |
| --- |
| typedef struct tagNETDEVAlarmRelatedData  {  [NETDEV\_STRUCT\_DATA\_INFO\_S](#_结构化数据信息) stStructDataInfo;  BYTE bRes[1024];  }NETDEV\_ALARM\_RELATED\_DATA\_S, \*LPNETDEV\_ALARM\_RELATED\_DATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stStructDataInfo | Info about alarm related structured data |
| byRes | Reserved field. |

### Basic info about alarm image

|  |
| --- |
| typedef struct tagNETDEVAlarmPicBaseInfo  {  CHAR szReference[NETDEV\_LEN\_260];  UINT32 udwAlarmType;  INT64 tTimeStamp;  UINT32 udwSeq;  UINT32 udwSourceID;  CHAR szSourceName[NETDEV\_LEN\_256];  CHAR szDeviceID[NETDEV\_LEN\_32];  CHAR szRelatedID[NETDEV\_LEN\_16];  BYTE byRes[256];  }NETDEV\_ALARM\_PIC\_BASE\_INFO\_S,\*LPNETDEV\_ALARM\_PIC\_BASE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szReference | Descriptive info |
| udwAlarmType | Alarm type. See [NETDEV\_ALARM\_TYPE\_E](#_告警类型枚举_1). |
| tTimeStamp | Alarm time. Number of seconds since 0’o clock, Jan. 1, 1970 |
| udwSeq | Alarm sequence number |
| udwSourceID | Alarm source ID. Applicable to VMS and IPC. |
| szSourceName | Alarm source name. Length: [1,63] |
| szDeviceID | Device ID. Fill in with the device’s GB code if connected via GB/T28181. Length: [1,32]. Applicable to IPC and VM. |
| szRelatedID | ID that relates image data with alarm event. Unique in the same camera. Length: 15 characters |
| byRes | Reserved field. |

### Alarm snapshot data

|  |
| --- |
| typedef struct tagNETDEVAlarmPicData  {  NETDEV\_ALARM\_PIC\_BASE\_INFO\_S stAlarmPicBaseInfo;  UINT32 udwImageNum;  [LPNETDEV\_STRUCT\_IMAGE\_INFO\_S](#_图像相关信息结构体) pstImageInfo;  BYTE byRes[512];  }NETDEV\_ALARM\_PIC\_DATA\_S,\*LPNETDEV\_ALARM\_PIC\_DATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stAlarmPicBaseInfo | Basic info about alarm image |
| udwImageNum | Number of images |
| pstImageInfo | Image info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### People counting alarm info

|  |
| --- |
| typedef struct tagNETDEVPeopleCountAlarmInfo  {  UINT32 udwType;  [NETDEV\_PEOPLE\_COUNT\_AREA\_RULE\_INFO\_S](#_人数统计区域规则统计数据信息) stAreaRuleInfo;  [NETDEV\_PEOPLE\_COUNT\_LINE\_RULE\_INFO\_S](#_人数统计绊线规则统计数据信息) stLineRuleInfo;  [NETDEV\_CROWD\_DENSITY\_INFO\_S](#_人员密度统计信息) stCrowdDensityInfo;  BYTE byRes[512];  }NETDEV\_PEOPLE\_COUNT\_ALARM\_INFO\_S,\*LPNETDEV\_PEOPLE\_COUNT\_ALARM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwType | People counting alarm type. See [NETDEV\_PEOPLE\_COUNT\_ALARM\_TYPE\_E](#_人数统计告警类型). |
| stAreaRuleInfo | Area people counting statistics |
| stLineRuleInfo | Tripwire people counting statistics |
| stCrowdDensityInfo | Crowd density statistics |
| byRes | Reserved field. |

### Area people counting statistics info

|  |
| --- |
| typedef struct tagNETDEVPeopleCountAreaRuleInfo  {  CHAR szReference[NETDEV\_LEN\_260];  INT64 tTimeStamp;  UINT32 udwSeq;  CHAR szDeviceID[NETDEV\_LEN\_32];  UINT32 udwChannelID;  UINT32 udwAreaNum;  [LPNETDEV\_AREA\_RULE\_COUNT\_DATA\_S](#_区域规则统计数据) pstAreaRuleCountDataList;  BYTE byRes[256];  }NETDEV\_PEOPLE\_COUNT\_AREA\_RULE\_INFO\_S,\*LPNETDEV\_PEOPLE\_COUNT\_AREA\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szReference | Descriptive info |
| tTimeStamp | Alarm time. Number of seconds since 0’o clock, Jan. 1, 1970 |
| udwSeq | Alarm sequence number |
| szDeviceID | Camera ID or area ID, device ID passed in for event subscription. Required when device ID is present in the event subscription interface. |
| udwChannelID | Channel ID. Starts from 0. |
| udwAreaNum | Number of area rules. Starts from 0. 0 means no rules. |
| pstAreaRuleCountDataList | Statistics of counting by detection area. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Tripwire people counting statistics info

|  |
| --- |
| typedef struct tagNETDEVPeopleCountLineRuleInfo  {  CHAR szReference[NETDEV\_LEN\_260];  INT64 tTimeStamp;  UINT32 udwSeq;  CHAR szDeviceID[NETDEV\_LEN\_32];  UINT32 udwChannelID;  UINT32 udwLineNum;  [LPNETDEV\_LINE\_RULE\_COUNT\_DATA\_S](#_伴线规则统计数据) pstLineRuleCountDataList;  BYTE byRes[256];  }NETDEV\_PEOPLE\_COUNT\_LINE\_RULE\_INFO\_S,\*LPNETDEV\_PEOPLE\_COUNT\_LINE\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szReference | Descriptive info |
| tTimeStamp | Reporting time. Number of seconds since 0’o clock, Jan. 1, 1970 |
| udwSeq | Sequence number of data to push |
| szDeviceID | Camera ID or area ID, device ID passed in for event subscription. Required when device ID is present in the event subscription interface. |
| udwChannelID | Channel ID. Starts from 0. |
| udwLineNum | Number of tripwire rules. Starts from 0. 0 means no rules. |
| pstLineRuleCountDataList | Statistics of counting by tripwire. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Crowd density statistics info

|  |
| --- |
| typedef struct tagNETDEVCrowdDensityInfo  {  CHAR szReference[NETDEV\_LEN\_260];  INT64 tTimeStamp;  UINT32 udwSeq;  UINT32 udwSrcID;  CHAR szSrcName[NETDEV\_LEN\_64];  UINT32 udwRelatedID;  [NETDEV\_CROWD\_DENSITY\_DATA\_S](#_人员密度统计数据) stCrowdDensityData;  BYTE byRes[256];  }NETDEV\_CROWD\_DENSITY\_INFO\_S,\*LPNETDEV\_CROWD\_DENSITY\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| szReference | Subscriber info |
| tTimeStamp | Alarm time. Number of seconds since 0’o clock, Jan. 1, 1970 |
| udwSeq | Sequence number of data to push |
| udwSrcID | Source ID |
| szSrcName | Source name. Length: [1,63] |
| udwRelatedID | Alarm event related ID |
| stCrowdDensityData | Crowd density data |
| byRes | Reserved field. |

### Area people counting statistics

|  |
| --- |
| typedef struct tagNETDEVAreaRuleCountData  {  UINT32 udwAreaID;  UINT32 udwObjectNum;  BYTE byRes[256];  }NETDEV\_AREA\_RULE\_COUNT\_DATA\_S,\*LPNETDEV\_AREA\_RULE\_COUNT\_DATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwAreaID | Detection area ID. Starts from 0. |
| udwObjectNum | Number of people present in the area |
| byRes | Reserved field. |

### Tripwire people counting statistics

|  |
| --- |
| typedef struct tagNETDEVLineRuleCountData  {  UINT32 udwLineID;  CHAR szBeginPassTime[NETDEV\_LEN\_64];  CHAR szEndPassTime[NETDEV\_LEN\_64];  UINT32 udwObjectIn;  UINT32 udwObjectOut;  BYTE byRes[256];  }NETDEV\_LINE\_RULE\_COUNT\_DATA\_S,\*LPNETDEV\_LINE\_RULE\_COUNT\_DATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwLineID | Tripwire ID. Starts from 0 |
| szBeginPassTime | Detection start time (YYYYMMDDHHMMSS, 24H format). String length range: [0,18] |
| szEndPassTime | Detection end time (YYYYMMDDHHMMSS, 24H format). String length range: [0,18] |
| udwObjectIn | Number of people that have entered during the set time |
| udwObjectOut | Number of people that have left during the set time |
| byRes | Reserved field. |

### Crowd density counting statistics

|  |
| --- |
| typedef struct tagNETDEVCrowdDensityData  {  UINT32 udwGroupID;  INT64 tBeginTime;  INT64 tEndTime;  UINT32 udwObjectIn;  UINT32 udwObjectOut;  [NETDEV\_CROWD\_DENSITY\_SUMMARY\_INFO\_S](#_人员密度统计汇总信息) stCrowdDensitySummaryInfo;  BYTE byRes[256];  }NETDEV\_CROWD\_DENSITY\_DATA\_S,\*LPNETDEV\_CROWD\_DENSITY\_DATA\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwGroupID | Counting group ID |
| tBeginTime | Counting start time (UTC). |
| tEndTime | Counting end time (UTC). |
| udwObjectIn | Number of people entered |
| udwObjectOut | Number of people left |
| stCrowdDensitySummaryInfo | Summary info |
| byRes | Reserved field. |

### Crowd density statistics summary info

|  |
| --- |
| typedef struct tagNETDEVCrowdDensitySummaryInfo  {  UINT32 udwTotalIn;  UINT32 udwTotalOut;  UINT32 udwAlarmThermal;  BYTE byRes[256];  }NETDEV\_CROWD\_DENSITY\_SUMMARY\_INFO\_S,\*LPNETDEV\_CROWD\_DENSITY\_SUMMARY\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwTotalIn | Total number of people that have entered |
| udwTotalOut | Total number of people that have left |
| udwAlarmThermal | Number of people as alarm threshold |
| byRes | Reserved field. |

### Crowd density group info

|  |
| --- |
| typedef struct tagNETDEVCrowdDensityGroupInfo  {  UINT32 udwGroupID;  CHAR szGroupName[NETDEV\_LEN\_64];  UINT32 udwStatisticalType;  UINT32 udwChannelRuleNum;  [LPNETDEV\_CHANNEL\_RULE\_INFO\_S](#_通道规则信息) pstChannelRuleInfoList;  [NETDEV\_CROWD\_DENSITY\_RULE\_INFO\_S](#_滞留规则信息) stCrowdDensityRuleInfo;  UINT32 udwReportInterval;  BYTE byRes[256];  }NETDEV\_CROWD\_DENSITY\_GROUP\_INFO\_S,\*LPNETDEV\_CROWD\_DENSITY\_GROUP\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwGroupID | Statistics group ID |
| szGroupName | Counting group name. Length range: [0,63]. |
| udwStatisticalType | Counting type. See [NETDEV\_CROWD\_DENSITY\_STATISTIC\_TYPE\_E](#_人员密度报警统计类型). |
| udwChannelRuleNum | Number of channel rules |
| pstChannelRuleInfoList | Channel rule list. Optional when udwChannelRuleNum is 0. Memory needs to be allocated dynamically by user. |
| stCrowdDensityRuleInfo | Crowd density rule info |
| udwReportInterval | Report interval (unit: second) |
| byRes | Reserved field. |

### Channel rule info

|  |
| --- |
| typedef struct tagNETDEVChannelRuleInfo  {  UINT32 udwChannelID;  UINT32 udwRuleNum;  UINT32 audwRuleIDList[NETDEV\_LEN\_64];  BYTE byRes[128];  }NETDEV\_CHANNEL\_RULE\_INFO\_S,\*LPNETDEV\_CHANNEL\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwChannelID | Channel ID |
| udwRuleNum | Number of rules |
| audwRuleIDList | Rule ID list |
| byRes | Reserved field. |

### Crowd density rule info

|  |
| --- |
| typedef struct tagNETDEVCrowdDensityRuleInfo  {  [NETDEV\_ALARM\_RULE\_INFO\_S](#_报警规则信息) stMinorAlarmRuleInfo;  [NETDEV\_ALARM\_RULE\_INFO\_S](#_报警规则信息) stMajorAlarmRuleInfo;  [NETDEV\_ALARM\_RULE\_INFO\_S](#_报警规则信息) stCriticalAlarmRuleInfo;  BYTE byRes[256];  }NETDEV\_CROWD\_DENSITY\_RULE\_INFO\_S,\*LPNETDEV\_CROWD\_DENSITY\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stMinorAlarmRuleInfo | Minor alarm rules |
| stMajorAlarmRuleInfo | Major alarm rules |
| stCriticalAlarmRuleInfo | Critical alarm rules |
| byRes | Reserved field. |

### Alarm rule Info

|  |
| --- |
| typedef struct tagNETDEVAlarmRuleInfo  {  BOOL bEnabled;  UINT32 udwAlarmThermal;  BYTE byRes[128];  }NETDEV\_ALARM\_RULE\_INFO\_S,\*LPNETDEV\_ALARM\_RULE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| bEnabled | Whether is enabled. FALSE: No TRUE: Yes |
| udwAlarmThermal | People counting alarm threshold |
| byRes | Reserved field. |

### Info on a single object

|  |
| --- |
| typedef struct tagNETDEVSingleObjectInfo  {  UINT32 udwObjectType;  [NETDEV\_FACE\_STRUCT\_INFO\_S](#_人脸信息) stFaceInfo;  [NETDEV\_PERSON\_STRUCT\_INFO\_S](#_人员信息) stPersonInfo;  [NETDEV\_NON\_MOTOR\_VEH\_INFO\_S](#_非机动车信息) stNonMotorVehInfo;  [NETDEV\_VEH\_INFO\_S](#_车辆信息) stVehInfo;  BYTE byRes[256];  }NETDEV\_SINGLE\_OBJECT\_INFO\_S,\*LPNETDEV\_SINGLE\_OBJECT\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwObjectType | Object type. See [NETDEV\_OBJECT\_TYPE\_E](#_目标类型枚举). |
| stFaceInfo | Face information |
| stPersonInfo | Person information |
| stNonMotorVehInfo | Non-motor vehicle information |
| stVehInfo | Vehicle information |
| byRes | Reserved field. |

### Fire detection alarm info

|  |
| --- |
| typedef struct tagNETDEVConflagrationAlarmInfoV30  {  [NETDEV\_ALARM\_PIC\_BASE\_INFO\_S](#_告警图片基础信息结构体) stConflagrationBaseInfo;  [NETDEV\_GEOLACATION\_INFO\_S](#_定位信息) stPTPositionInfo;  UINT32 udwNum;  [LPNETDEV\_CONFLAGRATION\_CHANNEL\_INFO\_S](#_火点告警通道信息结构体) pstChannelInfo;  FLOAT fZoom;  BYTE byRes[252];  }NETDEV\_CONFLAGRATION\_ALARM\_INFO\_V30\_S,\*LPNETDEV\_CONFLAGRATION\_ALARM\_INFO\_V30\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stConflagrationBaseInfo | Basic fire detection info |
| stPTPositionInfo | Position info |
| udwNum | Number of fire detection channels. |
| pstChannelInfo | Fire detection channel info list. Memory needs to be allocated dynamically. |
| fZoom | Zoom ratio |
| byRes | Reserved field. |

### Fire detection channel info

|  |
| --- |
| typedef struct tagNETDEVConflagrationChannelInfo  {  UINT32 udwChannelID;  FLOAT fLensView;  [NETDEV\_FIRE\_POINT\_LIST\_S](#_火点信息结构体) stFirePointList;  FLOAT fHorizontalFieldAngle;  FLOAT fVerticalFieldAngle;  BYTE byRes[248];  }NETDEV\_CONFLAGRATION\_CHANNEL\_INFO\_S,\*LPNETDEV\_CONFLAGRATION\_CHANNEL\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwChannelID | Channel ID. |
| fLensView | Lens’ field of view angle when fire is detected. Two decimal places. |
| stFirePointList | Position info of different fire points. |
| fHorizontalFieldAngle | Horizontal field of view angle when fire is detected. Two decimal places. |
| fVerticalFieldAngle | Vertical field of view angle when fire is detected. Two decimal places. |
| byRes | Reserved field. |

### Fire info list

|  |
| --- |
| typedef struct tagNETDEVFirePointList  {  UINT32 udwNum;  [LPNETDEV\_FIRE\_POINT\_S](#_火点位置信息结构体) pstFirePointsInfo;  UINT32 udwImageNum;  [LPNETDEV\_STRUCT\_IMAGE\_INFO\_S](#_图像相关信息结构体) pstImageInfo;  BYTE byRes[256];  }NETDEV\_FIRE\_POINT\_LIST\_S,\*LPNETDEV\_FIRE\_POINT\_LIST\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwNum | Number of fires. |
| pstFirePointsInfo | Fire position info list. Memory needs to be allocated dynamically. |
| udwImageNum | Number of images. |
| pstImageInfo | Image info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Fire position info

|  |
| --- |
| typedef struct tagNETDEVFirePoint  {  UINT32 udwId;  UINT32 udwNum;  [LPNETDEV\_POINT\_S](#_点坐标结构体) pstPoint;  BYTE byRes[128];  }NETDEV\_FIRE\_POINT\_S,\*LPNETDEV\_FIRE\_POINT\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwId | Fire ID |
| udwNum | Number of fires. |
| pstPoint | Coordinate information of the center position of the fire in the channel. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Smoking Detection Info

|  |
| --- |
| typedef struct tagNETDEVSmokeDetc  {  [NETDEV\_ALARM\_PIC\_BASE\_INFO\_S](#_告警图片基础信息结构体) stAlarmBaseInfo;  UINT32 udwChannelNum;  [LPNETDEV\_SMOKE\_DETC\_CHANNEL\_S](#_吸烟检测通道信息) pstSmokeDetcChannel;  BYTE byRes[256];  }NETDEV\_SMOKE\_DETC\_S,\*LPNETDEV\_SMOKE\_DETC\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stAlarmBaseInfo | Alarm basic info |
| udwChannelNum | Number of channels |
| pstSmokeDetcChannel | Smoking detection channel info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Smoking Detection Channel Info

|  |
| --- |
| typedef struct tagNETDEVSmokeDetcChannel  {  UINT32 udwChannelID;  UINT32 udwNum;  [LPNETDEV\_FIRE\_POINT\_S](#_火点位置信息结构体) pstFirePointList;  UINT32 udwImageNum;  [LPNETDEV\_STRUCT\_IMAGE\_INFO\_S](#_图像相关信息结构体) pstImageList;  BYTE byRes[256];  }NETDEV\_SMOKE\_DETC\_CHANNEL\_S,\*LPNETDEV\_SMOKE\_DETC\_CHANNEL\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwChannelID | Channel ID. |
| udwNum | Number of fires. |
| pstFirePointList | Fire position info list. Memory needs to be allocated dynamically. |
| udwImageNum | Number of images. |
| pstImageList | Image info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Temperature detection alarm info

|  |
| --- |
| typedef struct tagNETDEVTemperatureAlarmInfo  {  [NETDEV\_ALARM\_PIC\_BASE\_INFO\_S](#_告警图片基础信息结构体) stAlarmBaseInfo;  UINT32 udwRuleType;  UINT32 udwRuleId;  UINT32 udwReferenceRuleId;  UINT32 udwAlarmCondition;  UINT32 udwValueType;  FLOAT fAlarmValue;  FLOAT fThreshold;  UINT32 udwChannelNum;  LPNETDEV\_TEMPERATURE\_CHANNEL\_INFO\_S pstTemperatureChannelInfo;  BYTE byRes[256];  }NETDEV\_TEMPERATURE\_ALARM\_INFO\_S,\*LPNETDEV\_TEMPERATURE\_ALARM\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stAlarmBaseInfo | Alarm basic info |
| udwRuleType | Rule type, used to detect temperature alarms within the specified area. 1. Full screen detection. 2. Single rule detection (i.e., dot, line, polygon). 3. Temperature alarm rules, including single rule and comparison rule. 0xFF: invalid value. |
| udwRuleId | Rule ID. Required when RuleType is 2 or 3. |
| udwReferenceRuleId | Comparison rule ID. Required when RuleType is 3. |
| udwAlarmCondition | Temperature alarm triggering condition. 1: Lower than 2: Higher than 3: Match |
| udwValueType | Alarm value type. See [NETDEV\_TEMPERATURE\_TYPE\_E](#_温度告警值类型). |
| fAlarmValue | The alarm value generated by the current alarm type. When ValueType is 5, it represents the temperature change rate. Otherwise, it represents the temperature value with a precision of two decimal places. |
| fThreshold | When ValueType is 5, it represents the temperature change rate threshold. Otherwise, it represents the temperature threshold with a precision of two decimal places. |
| udwChannelNum | Number of channels. |
| pstTemperatureChannelInfo | Temperature detection channel info |
| byRes | Reserved field. |

### Temperature detection channel info

|  |
| --- |
| typedef struct tagNETDEVTemperatureChannelInfo  {  UINT32 udwChannelID;  UINT32 udwImageNum;  [LPNETDEV\_STRUCT\_IMAGE\_INFO\_S](#_图像相关信息结构体) pstImageList;  BYTE byRes[256];  }NETDEV\_SMOKE\_DETC\_CHANNEL\_S,\*LPNETDEV\_SMOKE\_DETC\_CHANNEL\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwChannelID | Channel ID. |
| udwImageNum | Number of images. |
| pstImageList | Image info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Fire detection info

|  |
| --- |
| typedef struct tagNETDEVConflagrationInfo  {  [NETDEV\_GEOLACATION\_INFO\_S](#_定位信息) stPTPositionInfo;  UINT32 udwNum;  [LPNETDEV\_CONFLAGRATION\_CHANNEL\_INFO\_S](#_火点告警通道信息结构体) pstChannelInfo;  BYTE byRes[256];  }NETDEV\_CONFLAGRATION\_INFO\_S,\*LPNETDEV\_CONFLAGRATION\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| stPTPositionInfo | Position info |
| udwNum | Number of fire detection channels. |
| pstChannelInfo | Fire detection channel info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Smoking info.

|  |
| --- |
| typedef struct tagNETDEVSmokingInfo  {  UINT32 udwChannelNum;  [LPNETDEV\_SMOKE\_DETC\_CHANNEL\_S](#_吸烟检测通道信息) pstSmokeDetcChannel;  BYTE byRes[256];  }NETDEV\_SMOKING\_INFO\_S,\*LPNETDEV\_SMOKING\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwChannelNum | Number of channels |
| pstSmokeDetcChannel | Smoking detection channel info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

### Temperature info

|  |
| --- |
| typedef struct tagNETDEVTemperatureInfo  {  UINT32 udwRuleType;  UINT32 udwRuleId;  UINT32 udwReferenceRuleId;  UINT32 udwAlarmCondition;  UINT32 udwValueType;  FLOAT fAlarmValue;  FLOAT fThreshold;  UINT32 udwChannelNum;  LPNETDEV\_TEMPERATURE\_CHANNEL\_INFO\_S pstTemperatureChannelInfo;  BYTE byRes[256];  }NETDEV\_TEMPERATURE\_INFO\_S,\*LPNETDEV\_TEMPERATURE\_INFO\_S; |

**Members:**

|  |  |
| --- | --- |
| Parameter | Description |
| udwRuleType | Rule type, used to detect temperature alarms within the specified area. 1. Full screen detection. 2. Single rule detection (i.e., dot, line, polygon). 3. Temperature alarm rules, including single rule and comparison rule. |
| udwRuleId | Rule ID. Required when RuleType is 2 or 3. |
| udwReferenceRuleId | Comparison rule ID. Required when RuleType is 3. |
| udwAlarmCondition | Temperature alarm triggering condition. 1: Lower than 2: Higher than 3: Match |
| udwValueType | Alarm value type. See [NETDEV\_TEMPERATURE\_TYPE\_E](#_温度告警值类型). |
| fAlarmValue | The alarm value generated by the current alarm type. When ValueType is 5, it represents the temperature change rate. Otherwise, it represents the temperature value with a precision of two decimal places. |
| fThreshold | When ValueType is 5, it represents the temperature change rate threshold. Otherwise, it represents the temperature threshold with a precision of two decimal places. |
| udwChannelNum | Number of channels. |
| pstTemperatureChannelInfo | Temperature detection channel info. Memory needs to be allocated dynamically. |
| byRes | Reserved field. |

## Definition of Enumerations

### Enumeration of access protocols

typedef enum tagNETDEVLoginProto

{

NETDEV\_LOGIN\_PROTO\_ONVIF = 0, /\* Onvif\*/

NETDEV\_LOGIN\_PROTO\_PRIVATE = 1 /\* Private protocol\*/

}NETDEV\_LOGIN\_PROTO\_E;

### Enumeration of exception message types

typedef enum tagNETDEVException

{

NETDEV\_EXCEPTION\_REPORT\_REMUXING\_FINISH = 284, /\* Repackaging finished \*/

NETDEV\_EXCEPTION\_REPORT\_VOD\_END = 300, /\* Playback ended \*/

NETDEV\_EXCEPTION\_REPORT\_VOD\_ABEND = 301, /\* Playback abnormal \*/

NETDEV\_EXCEPTION\_REPORT\_BACKUP\_END = 302, /\* Backup ended \*/

NETDEV\_EXCEPTION\_REPORT\_BACKUP\_DISC\_OUT = 303, /\* HDD removed \*/

NETDEV\_EXCEPTION\_REPORT\_BACKUP\_DISC\_FULL = 304, /\* HDD full \*/

NETDEV\_EXCEPTION\_REPORT\_BACKUP\_ABEND = 305, /\* Backup failed for other causes \*/

NETDEV\_EXCEPTION\_EXCHANGE = 0x8000, /\* Exception during user interaction (keep-alive timeout)\*/

NETDEV\_EXCEPTION\_REPORT\_ALARM\_INTERRUPT = 0x8001, /\* Alarm reporting ended abnormally Keep-alive failed or persistent connection disconnected\*/

NETDEV\_EXCEPTION\_REPORT\_MAX, /\* Maximum value \*/

NETDEV\_EXCEPTION\_REPORT\_NOT\_VALID\_PERIOD, /\* Not within valid period \*/

NETDEV\_EXCEPTION\_REPORT\_NOT\_VALID\_TIME, /\* Not within valid period \*/

NETDEV\_EXCEPTION\_REPORT\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_EXCEPTION\_TYPE\_E;

### Enumeration of device types

typedef enum tagNETDEVDeviceType

{

NETDEV\_DTYPE\_UNKNOWN = 0, /\* Unknown type \*/

NETDEV\_DTYPE\_IPC = 1, /\* IPC range \*/

NETDEV\_DTYPE\_IPC\_FISHEYE = 2, /\* Fisheye camera (with fisheye lens + dewarping) \*/

NETDEV\_DTYPE\_IPC\_ECONOMIC\_FISHEYE = 3, /\* Fisheye camera (with fisheye lens or wide angle lens without dewarping \*/

NETDEV\_DTYPE\_IPC\_ACS = 4, /\* Face recognition terminal \*/

NETDEV\_DTYPE\_NVR = 101, /\* NVR range \*/

NETDEV\_DTYPE\_NVR\_BACKUP = 102, /\* NVR backup server \*/

NETDEV\_DTYPE\_HNVR = 103, /\* Hybrid NVR \*/

NETDEV\_DTYPE\_DC = 201, /\* DC \*/

NETDEV\_DTYPE\_DC\_ADU = 202, /\* ADU \*/

NETDEV\_DTYPE\_EC = 301, /\* EC \*/

NETDEV\_DTYPE\_VMS = 501, /\* VMS \*/

NETDEV\_DTYPE\_FG = 601, /\* FG \*/

NETDEV\_DTYPE\_IPM = 701, /\* IPM \*/

NETDEV\_DTYPE\_EDU = 801, /\* EDU \*/

NETDEV\_DTYPE\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_DEVICE\_TYPE\_E;

### Enumeration of stream types

typedef enum tagNETDEVLiveStreamIndex

{

NETDEV\_LIVE\_STREAM\_INDEX\_MAIN = 0, /\* Main stream \*/

NETDEV\_LIVE\_STREAM\_INDEX\_AUX = 1, /\* Sub stream \*/

NETDEV\_LIVE\_STREAM\_INDEX\_THIRD = 2, /\* Third stream \*/

NETDEV\_LIVE\_STREAM\_INDEX\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_LIVE\_STREAM\_INDEX\_E;

### Enumeration of media transport protocols

typedef enum tagNETDEVProtocal

{

NETDEV\_TRANSPROTOCAL\_RTPUDP = 0, /\* UDP \*/

NETDEV\_TRANSPROTOCAL\_RTPTCP = 1 /\* TCP \*/

}NETDEV\_PROTOCAL\_E;

### Enumeration of image fluency

typedef enum tagNetDEVPictureFluency

{

NETDEV\_PICTURE\_REAL = 0, /\* Real-time priority \*/

NETDEV\_PICTURE\_FLUENCY = 1, /\* Fluency priority \*/

NETDEV\_PICTURE\_BALANCE\_NEW = 3, /\* Balanced \*/

NETDEV\_PICTURE\_RTMP\_FLUENCY = 4, /\* RTMP fluency priority \*/

NETDEV\_PICTURE\_USER\_DEFINED = 5, /\* Custom-configurable buffered frames \*/

NETDEV\_PICTURE\_NETADJUST = 6, /\* Network jitter adaptation mode \*/

NETDEV\_PICTURE\_FLUENCY\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_PICTURE\_FLUENCY\_E;

### Enumeration of stream modes

typedef enum tagNETDEVStreamMode

{

NETDEV\_STREAM\_MODE\_ALL = 0x0000, /\* audio + video \*/

NETDEV\_STREAM\_MODE\_VIDEO = 0x8000 /\* video only \*/

}NETDEV\_STREAM\_MODE\_E;

### Enumeration of transport types

typedef enum tagNETDEVTransType

{

NETDEV\_TRANS\_TYPE\_FORWORD = 0, /\* Forward via VMS \*/

NETDEV\_TRANS\_TYPE\_STRAIGHT = 1 /\* Direct connection with downlink devices \*/

}NETDEV\_TRANS\_TYPE\_E;

### Enumeration of streaming protocols

typedef enum tagNETDEVStartStreamProt

{

NETDEV\_START\_STREAM\_PROT\_UDP =1,

NETDEV\_START\_STREAM\_PROT\_TCP =2,

NETDEV\_START\_STREAM\_PROT\_RTSP =3,

NETDEV\_START\_STREAM\_PROT\_HTTP =4,

NETDEV\_START\_STREAM\_PROT\_INVALID =0xff

}NETDEV\_START\_STREAM\_PROT\_E;

### Enumeration of media file formats

typedef enum tagNETDEVMediaFileFormat

{

NETDEV\_MEDIA\_FILE\_MP4 = 0, /\* MP4 (audio+video) \*/

NETDEV\_MEDIA\_FILE\_TS = 1, /\* TS (audio+video) \*/

NETDEV\_MEDIA\_FILE\_MP4\_ADD\_TIME = 2, /\* MP4 (audio+video), filename including timestamp \*/

NETDEV\_MEDIA\_FILE\_TS\_ADD\_TIME = 3, /\* TS (audio+video), filename including timestamp \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_TS = 4, /\* TS (audio only) \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_MP4 = 5, /\* MP4 (audio only) \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_TS\_ADD\_TIME = 6, /\* TS (audio only), filename including timestamp \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_MP4\_ADD\_TIME = 7, /\* MP4 (audio only), filename including timestamp \*/

NETDEV\_MEDIA\_FILE\_MP4\_ADD\_RCD\_TIME = 8, /\* MP4 (audio+video), filename including start and end times \*/

NETDEV\_MEDIA\_FILE\_TS\_ADD\_RCD\_TIME = 9, /\* TS (audio+video), filename including start and end times \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_MP4\_ADD\_RCD\_TIME = 10, /\* MP4 (audio only), filename including start and end times \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_TS\_ADD\_RCD\_TIME = 11, /\* TS (audio only), filename including start and end times \*/

NETDEV\_MEDIA\_FILE\_VIDEO\_MP4\_ADD\_RCD\_TIME = 12, /\* MP4 (video only) \*/

NETDEV\_MEDIA\_FILE\_VIDEO\_TS\_ADD\_RCD\_TIME = 13, /\* TS (video only) \*/

NETDEV\_MEDIA\_FILE\_AVI = 14, /\* AVI (audio+video) \*/

NETDEV\_MEDIA\_FILE\_AVI\_ADD\_TIME = 15, /\* AVI (audio+video), filename including start and end times \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_AVI = 16, /\* AVI (audio only) \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_AVI\_ADD\_TIME = 17, /\* AVI (audio only), filename including start and end times \*/

NETDEV\_MEDIA\_FILE\_AVI\_ADD\_RCD\_TIME = 18, /\* AVI (audio+video), filename including start and end times \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_AVI\_ADD\_RCD\_TIME = 19, /\* AVI (audio only), filename including start and end times \*/

NETDEV\_MEDIA\_FILE\_VIDEO\_AVI\_ADD\_RCD\_TIME = 20, /\* AVI (video only) \*/

NETDEV\_MEDIA\_FILE\_INVALID

}NETDEV\_MEDIA\_FILE\_FORMAT\_E;

### Enumeration of snapshot image formats

typedef enum tagNETDEVPictureFormat

{

NETDEV\_PICTURE\_BMP = 0, /\* BMP format \*/

NETDEV\_PICTURE\_JPG = 1, /\* JPG format\*/

NETDEV\_PICTURE\_INVALID

}NETDEV\_PICTURE\_FORMAT\_E;

### Enumeration of audio formats

typedef enum tagNetDEVWaveFormatInfo

{

NETDEV\_WAVE\_FORMAT\_PCM8M16 = 0, /\* PCM, 8K sampling rate, mono, 16 bits\*/

NETDEV\_WAVE\_FORMAT\_PCM16M16 = 1, /\* PCM,16K sampling rate, mono, 16 bits\*/

NETDEV\_WAVE\_FORMAT\_PCM441M16 = 2, /\* PCM, 44.1K sampling rate, mono, 16 bits\*/

NETDEV\_WAVE\_FORMAT\_PCM11025M16 = 3, /\* PCM, 11.025K sampling rate, mono, 16 bits\*/

NETDEV\_WAVE\_FORMAT\_PCM8\_RIGHT = 4, /\* PCM, 8K sampling rate, right channel\*/

NETDEV\_WAVE\_FORMAT\_PCM\_NORMAL = 5, /\* PCM, normal mode\*/

NETDEV\_WAVE\_FORMAT\_PCM48000M16 = 6, /\* PCM, 48.0K sampling rate, mono (M), 16 bits\*/

NETDEV\_WAVE\_FORMAT\_UNKNOWN

}NETDEV\_WAVE\_FORMAT\_INFO\_E;

### Enumeration of audio bit widths

typedef enum tagNetDevAudioSampleFormatType

{

NETDEV\_AUDIO\_SAMPLE\_FMT\_NONE = -1,

NETDEV\_AUDIO\_SAMPLE\_FMT\_U8 = 0, /\* unsigned integer\*/

NETDEV\_AUDIO\_SAMPLE\_FMT\_S16 = 1, /\* signed integer\*/

NETDEV\_AUDIO\_SAMPLE\_FMT\_S32 = 2, /\* signed integer\*/

NETDEV\_AUDIO\_SAMPLE\_FMT\_FLT = 3, /\* floating\*/

NETDEV\_AUDIO\_SAMPLE\_FMT\_DBL = 4 /\* double\*/

}NETDEV\_AUDIO\_SAMPLE\_FORMAT\_E;

### Enumeration of media stream formats

typedef enum tagNETDEVMediaDataFormat

{

NETDEV\_MEDIA\_DATA\_TS = 0, /\* TS \*/

NETDEV\_MEDIA\_DATA\_RTP\_ES = 3 /\* RTP + ES \*/

}NETDEV\_MEDIA\_DATA\_FORMAT\_E;

### Enumeration of audio encoding formats

typedef enum tagNetDEVAudioEncodeType

{

NETDEV\_AUDIO\_ENCODE\_G711U = 0, /\* G.711U \*/

NETDEV\_AUDIO\_ENCODE\_MP2 = 1, /\* MP2 \*/

NETDEV\_AUDIO\_ENCODE\_AAC = 2, /\* AAC \*/

NETDEV\_AUDIO\_ENCODE\_G711A = 3, /\* G.711A \*/

NETDEV\_AUDIO\_ENCODE\_PCM = 4, /\* PCM \*/

NETDEV\_AUDIO\_ENCODE\_NULL = 5 /\* None \*/

}NETDEV\_AUDIO\_ENCODE\_TYPE\_E;

### Enumeration of video frame types

typedef enum tagNETDEVVideoFrameType

{

NETDEV\_VIDEO\_FRAME\_I = 0, /\* I-frame \*/

NETDEV\_VIDEO\_FRAME\_P = 1, /\* P-frame \*/

NETDEV\_VIDEO\_FRAME\_B = 2 /\* B-frame \*/

}NETDEV\_VIDEO\_FRAME\_TYPE\_E;

### Enumeration of video compression formats

typedef enum tagNETDEVVideoCodeType

{

NETDEV\_VIDEO\_CODE\_MJPEG = 0, /\* MJPEG \*/

NETDEV\_VIDEO\_CODE\_H264 = 1, /\* H.264 \*/

NETDEV\_VIDEO\_CODE\_H265 = 2, /\* H.265 \*/

NETDEV\_VIDEO\_CODE\_INVALID

}NETDEV\_VIDEO\_CODE\_TYPE\_E;

### Enumeration of metadata display types

typedef enum tagNETDEVIVAShowRule

{

NETDEV\_IVA\_SHOW\_RULE = 0x00000001, /\* Show rules \*/

NETDEV\_IVA\_SHOW\_RESULT\_TOUTH\_RULE = 0x00000002, /\* Show bounding boxes of objects that have triggered rules \*/

NETDEV\_IVA\_SHOW\_RESULT\_UNTOUTH\_RULE = 0x00000004, /\* Shows bounding boxes of objects that haven’t triggered rules \*/

NETDEV\_IVA\_SHOW\_TRACK = 0x00000008 /\* Shows the track of bounding boxes \*/

}NETDEV\_IVA\_SHOW\_RULE\_E;

### Enumeration of video display scales

typedef enum tagNETDEVRenderScale

{

NETDEV\_RENDER\_SCALE\_FULL = 0, /\* Fill the entire window \*/

NETDEV\_RENDER\_SCALE\_PROPORTION = 1, /\* Display by proportion \*/

NETDEV\_RENDER\_SCALE\_INVALID = 0xFF

}NETDEV\_RENDER\_SCALE\_E;

### Enumeration of recording storage types

typedef enum tagNETDEVPLanStoreType

{

NETDEV\_STORE\_TYPE\_COMMON = 0, /\* Normal recording \*/

NETDEV\_STORE\_TYPE\_DIGITIALINPUT = 1, /\* Triggered by alarm input \*/

NETDEV\_STORE\_TYPE\_MANUL = 2, /\* Manual recording \*/

NETDEV\_STORE\_TYPE\_AUDIODETECT = 3, /\* Triggered by audio detection alarm\*/

NETDEV\_STORE\_TYPE\_MOTION = 4, /\* Triggered by motion detection alarm \*/

NETDEV\_STORE\_TYPE\_DIGITALINPUT = 5, /\* Triggered by digital input event \*/

NETDEV\_STORE\_TYPE\_FACEDETECT = 6, /\* Triggered by face detection alarm \*/

NETDEV\_STORE\_TYPE\_VIDEO\_LOSS = 7, /\* Triggered by video loss alarm \*/

NETDEV\_STORE\_TYPE\_LINEDETECT = 8, /\* Triggered by cross line detection alarm \*/

NETDEV\_STORE\_TYPE\_FIELDDETECT = 9, /\* Triggered by intrusion detection alarm \*/

NETDEV\_STORE\_TYPE\_FOCUSDETECT = 10, /\* Triggered by defocus detection alarm \*/

NETDEV\_STORE\_TYPE\_SCENECHANGE = 11, /\* Triggered by scene change detection alarm \*/

NETDEV\_STORE\_TYPE\_ALARM = 12, /\* Alarm \*/

NETDEV\_STORE\_TYPE\_ALARM\_AND\_MOTION = 13, /\* Triggered by motion detection and alarm \*/

NETDEV\_STORE\_TYPE\_ALARM\_OR\_MOTION = 14, /\* Triggered by motion detection or alarm \*/

NETDEV\_STORE\_TYPE\_CAMERA\_DISCONNECT = 15, /\* Triggered camera offline alarm \*/

NETDEV\_STORE\_TYPE\_THIRD\_STREAM = 16, /\* Third-stream recording \*/

NETDEV\_STORE\_TYPE\_EVENT\_ALL\_ALARM = 17, /\* Event recording, including all alarm types \*/

NETDEV\_STORE\_TYPE\_EVENT\_ALL\_TYPE = 18, /\* Event recording, including all recording types \*/

NETDEV\_PLAN\_STORE\_TYPE\_EVENT\_WITHOUT\_RESUME = 19, /\* Recording of events that cannot be resumed \*/

NETDEV\_STORE\_TYPE\_SMART\_TRACK = 20, /\* Auto track \*/

NETDEV\_STORE\_TYPE\_URGENT\_BELL = 21, /\* emergency bell \*/

NETDEV\_STORE\_TYPE\_REMAIN\_ARTICLE = 22, /\* Object left behind \*/

NETDEV\_STORE\_TYPE\_MOVE\_ARTICLE = 23, /\* Object removed \*/

NETDEV\_STORE\_TYPE\_SMART\_RECORD = 24, /\* All smart alarms \*/

NETDEV\_PLAN\_STORE\_TYPE\_MAX = 25, /\* Maximum value of storage type \*/

NETDEV\_STORE\_TYPE\_HUMAN\_DETECT = 26, /\* Human detection \*/

NETDEV\_STORE\_TYPE\_ELEVATOR\_ENTRANCE\_DETECT = 27, /\* Elevator entrance detection \*/

NETDEV\_STORE\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_STORE\_TYPE\_E;

### Enumeration of recording storage locations

typedef enum tagNETDEVRecordLocation

{

NETDEV\_RECORD\_LOCATION\_ALL = 0, /\* All \*/

NETDEV\_RECORD\_LOCATION\_VMS = 1, /\* VMS \*/

NETDEV\_RECORD\_LOCATION\_NVR = 2, /\* NVR \*/

NETDEV\_RECORD\_LOCATION\_BACKUP = 3, /\* Backup \*/

NETDEV\_RECORD\_LOCATION\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_RECORD\_LOCATION\_E;

### Enumeration of recording search locations

typedef enum tagNETDEVVideoPosition

{

NETDEV\_POSITION\_LOCAL = 1, /\* Device recording \*/

NETDEV\_POSITION\_CHANNEL = 2, /\* Channel recording \*/

NETDEV\_POSITION\_INVALID

}NETDEV\_VIDEO\_POSITION\_E;

### Enumeration of recording status

typedef enum tagNETDEVVideoStatus

{

NETDEV\_VIDEO\_NONE = 0, /\* No recording \*/

NETDEV\_VIDEO\_EVENT = 1, /\* Event recording \*/

NETDEV\_VIDEO\_NORMAL = 2, /\* Normal recording \*/

NETDEV\_VIDEO\_INVALID

}NETDEV\_VIDEO\_STATUS\_E;

### Enumeration of recording download speeds

typedef enum tagNETDEVDownloadSpeed

{

NETDEV\_DOWNLOAD\_SPEED\_ONE = 0, /\* 1x \*/

NETDEV\_DOWNLOAD\_SPEED\_TWO = 1, /\* 2x \*/

NETDEV\_DOWNLOAD\_SPEED\_FOUR = 2, /\* 4x \*/

NETDEV\_DOWNLOAD\_SPEED\_EIGHT = 3, /\* 8x \*/

NETDEV\_DOWNLOAD\_SPEED\_TWO\_IFRAME = 4, /\* 2x I frame \*/

NETDEV\_DOWNLOAD\_SPEED\_FOUR\_IFRAME = 5, /\* 4x I frame \*/

NETDEV\_DOWNLOAD\_SPEED\_EIGHT\_IFRAME = 6, /\* 8x I frame \*/

NETDEV\_DOWNLOAD\_SPEED\_HALF = 7, /\* 1/2x \*/

NETDEV\_DOWNLOAD\_SPEED\_FORTY = 8, /\* 40x \*/

NETDEV\_DOWNLOAD\_SEPPD\_SIXTEEN = 9, /\* 16x \*/

NETDEV\_DOWNLOAD\_SPEED\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_E\_DOWNLOAD\_SPEED\_E;

### Enumeration of play/download speeds

typedef enum tagNETDEVVodPlayStatus

{

NETDEV\_PLAY\_STATUS\_16\_BACKWARD = 0, /\* 16x backward \*/

NETDEV\_PLAY\_STATUS\_8\_BACKWARD = 1, /\* 8x backward \*/

NETDEV\_PLAY\_STATUS\_4\_BACKWARD = 2, /\* 4x backward \*/

NETDEV\_PLAY\_STATUS\_2\_BACKWARD = 3, /\* 2x backward \*/

NETDEV\_PLAY\_STATUS\_1\_BACKWARD = 4, /\* 1x backward \*/

NETDEV\_PLAY\_STATUS\_HALF\_BACKWARD = 5, /\* 1/2x backward \*/

NETDEV\_PLAY\_STATUS\_QUARTER\_BACKWARD = 6, /\* 1/4x backward \*/

NETDEV\_PLAY\_STATUS\_QUARTER\_FORWARD = 7, /\* 1/4x forward \*/

NETDEV\_PLAY\_STATUS\_HALF\_FORWARD = 8, /\* 1/2x forward \*/

NETDEV\_PLAY\_STATUS\_1\_FORWARD = 9, /\* 1x forward \*/

NETDEV\_PLAY\_STATUS\_2\_FORWARD = 10, /\* 2x forward \*/

NETDEV\_PLAY\_STATUS\_4\_FORWARD = 11, /\* 4x forward \*/

NETDEV\_PLAY\_STATUS\_8\_FORWARD = 12, /\* 8x forward \*/

NETDEV\_PLAY\_STATUS\_16\_FORWARD = 13, /\* 16x forward \*/

NETDEV\_PLAY\_STATUS\_2\_FORWARD\_IFRAME = 14, /\* 2x forward (I frame) \*/

NETDEV\_PLAY\_STATUS\_4\_FORWARD\_IFRAME = 15, /\* 4x forward (I frame) \*/

NETDEV\_PLAY\_STATUS\_8\_FORWARD\_IFRAME = 16, /\* 8x forward (I frame) \*/

NETDEV\_PLAY\_STATUS\_16\_FORWARD\_IFRAME = 17, /\* 16x forward (I frame) \*/

NETDEV\_PLAY\_STATUS\_2\_BACKWARD\_IFRAME = 18, /\* 2x backward (I frame) \*/

NETDEV\_PLAY\_STATUS\_4\_BACKWARD\_IFRAME = 19, /\* 4x backward (I frame) \*/

NETDEV\_PLAY\_STATUS\_8\_BACKWARD\_IFRAME = 20, /\* 8x backward (I frame) \*/

NETDEV\_PLAY\_STATUS\_16\_BACKWARD\_IFRAME = 21, /\* 16x backward (I frame) \*/

NETDEV\_PLAY\_STATUS\_INTELLIGENT\_FORWARD = 22, /\* Smart forward \*/

NETDEV\_PLAY\_STATUS\_1\_FRAME\_FORWD = 23, /\* Forward frame by frame\*/

NETDEV\_PLAY\_STATUS\_1\_FRAME\_BACK = 24, /\* Backward frame by frame \*/

NETDEV\_PLAY\_STATUS\_40\_FORWARD = 25, /\* 40x forward \*/

NETDEV\_PLAY\_STATUS\_32\_FORWARD\_IFRAME = 26, /\* 32x forward (I frame) \*/

NETDEV\_PLAY\_STATUS\_32\_BACKWARD\_IFRAME = 27, /\* 32x backward (I frame) \*/

NETDEV\_PLAY\_STATUS\_64\_FORWARD\_IFRAME = 28, /\* 64x forward (I frame) \*/

NETDEV\_PLAY\_STATUS\_64\_BACKWARD\_IFRAME = 29, /\* 64x backward (I frame) \*/

NETDEV\_PLAY\_STATUS\_128\_FORWARD\_IFRAME = 30, /\* 128x forward (I frame) \*/

NETDEV\_PLAY\_STATUS\_128\_BACKWARD\_IFRAME = 31, /\* 128x backward (I frame) \*/

NETDEV\_PLAY\_STATUS\_256\_FORWARD\_IFRAME = 32, /\* 256x forward (I frame) \*/

NETDEV\_PLAY\_STATUS\_256\_BACKWARD\_IFRAME = 33, /\* 256x backward (I frame) \*/

NETDEV\_PLAY\_STATUS\_32\_FORWARD = 34, /\* 32x forward \*/

NETDEV\_PLAY\_STATUS\_32\_BACKWARD = 35, /\* 32x backward \*/

NETDEV\_PLAY\_STATUS\_INVALID

}NETDEV\_VOD\_PLAY\_STATUS\_E;

### Enumeration of PTZ commands

typedef enum tagNETDEVPtzCmdEnum

{

NETDEV\_PTZ\_IRISCLOSE\_STOP =0x0101, /\* Stop closing iris \*/

NETDEV\_PTZ\_IRISCLOSE =0x0102, /\* Close iris \*/

NETDEV\_PTZ\_IRISOPEN\_STOP =0x0103, /\* Stop opening iris \*/

NETDEV\_PTZ\_IRISOPEN =0x0104, /\* Open iris \*/

NETDEV\_PTZ\_FOCUSNEAR\_STOP =0x0201, /\* Stop focusing near \*/

NETDEV\_PTZ\_FOCUSNEAR =0x0202, /\* Focus near \*/

NETDEV\_PTZ\_FOCUSFAR\_STOP =0x0203, /\* Stop focusing far \*/

NETDEV\_PTZ\_FOCUSFAR =0x0204, /\* Focus far \*/

NETDEV\_PTZ\_ZOOMTELE\_STOP = 0x0301, /\* Stop zooming wide \*/

NETDEV\_PTZ\_ZOOMTELE = 0x0302, /\* Zoom wide \*/

NETDEV\_PTZ\_ZOOMWIDE\_STOP = 0x0303, /\* Stop zooming tele \*/

NETDEV\_PTZ\_ZOOMWIDE = 0x0304, /\* Zoom tele \*/

NETDEV\_PTZ\_TILTUP = 0x0402, /\* Tilt up \*/

NETDEV\_PTZ\_TILTDOWN = 0x0404, /\* Tilt down \*/

NETDEV\_PTZ\_PANRIGHT = 0x0502, /\* Pan right \*/

NETDEV\_PTZ\_PANLEFT = 0x0504, /\* Pan left \*/

NETDEV\_PTZ\_LEFTUP = 0x0702, /\* Move upper left \*/

NETDEV\_PTZ\_LEFTDOWN = 0x0704, /\* Move lower left \*/

NETDEV\_PTZ\_RIGHTUP = 0x0802, /\* Move upper right \*/

NETDEV\_PTZ\_RIGHTDOWN = 0x0804, /\* Move lower right \*/

NETDEV\_PTZ\_ALLSTOP = 0x0901, /\* All stop \*/

NETDEV\_PTZ\_FOCUS\_AND\_IRIS\_STOP = 0x0907, /\* Stop focus and iris\*/

NETDEV\_PTZ\_MOVE\_STOP = 0x0908, /\* Stop moving \*/

NETDEV\_PTZ\_ZOOM\_STOP = 0x0909, /\* Stop zoom \*/

NETDEV\_PTZ\_TRACKCRUISE = 0x1001, /\* Start recorded patrol \*/

NETDEV\_PTZ\_TRACKCRUISESTOP = 0x1002, /\* Stop recorded patrol \*/

NETDEV\_PTZ\_TRACKCRUISEREC = 0x1003, /\* Start recording patrol route \*/

NETDEV\_PTZ\_TRACKCRUISERECSTOP = 0x1004, /\* Stop recording patrol route \*/

NETDEV\_PTZ\_TRACKCRUISEADD = 0x1005, /\* Add a patrol route \*/

NETDEV\_PTZ\_TRACKCRUISEDEL = 0x1006, /\* Delete a patrol route \*/

NETDEV\_PTZ\_AREAZOOMIN = 0x1101, /\* Drag to zoom in\*/

NETDEV\_PTZ\_AREAZOOMOUT = 0x1102, /\* Drag to zoom out \*/

NETDEV\_PTZ\_AREAZOOM3D = 0x1103, /\* 3D positioning \*/

NETDEV\_PTZ\_BRUSHON = 0x0A01, /\* Wiper on \*/

NETDEV\_PTZ\_BRUSHOFF = 0x0A02, /\* Wiper off \*/

NETDEV\_PTZ\_LIGHTON = 0x0B01, /\* Light on \*/

NETDEV\_PTZ\_LIGHTOFF = 0x0B02, /\* Light off \*/

NETDEV\_PTZ\_HEATON = 0x0C01, /\* Heater on \*/

NETDEV\_PTZ\_HEATOFF = 0x0C02, /\* Heater off \*/

NETDEV\_PTZ\_SNOWREMOINGON = 0x1301, /\* Snow removal on \*/

NETDEV\_PTZ\_SNOWREMOINGOFF = 0x1302, /\* Snow removal off \*/

NETDEV\_PTZ\_INFRAREDON = 0x0D01, /\* IR on \*/

NETDEV\_PTZ\_INFRAREDOFF = 0x0D02, /\* IR off \*/

NETDEV\_PTZ\_SELF\_CHECKING = 0x1801, /\* self check: When dwSpeed is 1, perform PTZ self-check. When dwSpeed is 0, reboot the camera to restore the default physical parameters of PTZ. \*/

NETDEV\_PTZ\_INVALID

} NETDEV\_PTZ\_E;

### PTZ auto guard configuration commands

typedef enum tagNETDEVPTZGuardCmd

{

NETDEV\_PTZ\_GUARD\_CMD\_GET = 0, /\* Get auto guard info of specified channel\*/

NETDEV\_PTZ\_GUARD\_CMD\_SET = 1 /\* Set auto guard info of specified channel\*/

}NETDEV\_PTZ\_GUARD\_CMD\_E;

### PTZ auto guard modes

typedef enum tagNETDEVPTZGuardMode

{

NETDEV\_PTZ\_GUARD\_MODE\_PRESET = 0, /\* Preset mode\*/

NETDEV\_PTZ\_GUARD\_MODE\_CRUISE = 1, /\* Preset patrol mode\*/

NETDEV\_PTZ\_GUARD\_MODE\_TRACK = 2 /\* Recorded patrol mode\*/

}NETDEV\_PTZ\_GUARD\_MODE\_E;

### PTZ home position commands

typedef enum tagNETDEVPtzHomePositionCmd

{

NETDEV\_PTZ\_SET\_HOMEPOSITION = 0, /\* Set home position \*/

NETDEV\_PTZ\_GOTO\_HOMEPOSITION = 1 /\* Go to home position \*/

}NETDEV\_PTZ\_HOMEPOSITIONCMD\_E;

### Focus status

typedef enum tagNETDEVPTZMoveStatus

{

NETDEV\_PTZ\_MOVE\_STATUS\_IDLE = 0, /\* Free \*/

NETDEV\_PTZ\_MOVE\_STATUS\_MOVING = 1, /\* Moving \*/

NETDEV\_PTZ\_MOVE\_STATUS\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_PTZ\_MOVE\_STATUS\_E;

### Enumeration of patrol statuses

typedef enum tagNETDEVPTZTrackStatus

{

NETDEV\_PTZ\_TRACK\_STATUS\_FREE = 0, /\* Free (not patrolling, not recording a patrol route)\*/

NETDEV\_PTZ\_TRACK\_STATUS\_RUN = 1, /\* Patrolling\*/

NETDEV\_PTZ\_TRACK\_STATUS\_START\_RECORD = 2 /\* Recording a patrol route\*/

}NETDEV\_PTZ\_TRACK\_STATUS\_E;

### Enumeration of PTZ patrol operation

typedef enum tagNETDEVPTZTRACKCmd

{

NETDEV\_PTZ\_TRACKCRUISE\_START = 0, /\* Start recorded patrol \*/

NETDEV\_PTZ\_TRACKCRUISE\_STOP = 1, /\* Stop recorded patrol \*/

NETDEV\_PTZ\_TRACKCRUISE\_REC = 2, /\* Start recording a patrol route \*/

NETDEV\_PTZ\_TRACKCRUISE\_RECSTOP = 3, /\* Stop recording a patrol route \*/

NETDEV\_PTZ\_TRACKCRUISE\_ADD = 4, /\* Add a recorded patrol route. Currently not available to VMS server. The patrol route ID is fixed to 16 \*/

NETDEV\_PTZ\_TRACKCRUISE\_DEL = 5 /\* Delete a recorded patrol route \*/

}NETDEV\_PTZ\_TRACKCMD\_E;

### Enumeration of PTZ preset operation commands

typedef enum tagNETDEVPtzPresetCmd

{

NETDEV\_PTZ\_SET\_PRESET = 0, /\* Set preset \*/

NETDEV\_PTZ\_CLE\_PRESET = 1, /\* Delete preset \*/

NETDEV\_PTZ\_GOTO\_PRESET = 2 /\* Go to preset \*/

}NETDEV\_PTZ\_PRESETCMD\_E;

### Enumeration of PTZ patrol operation

typedef enum tagNETDEVPTZCruiseCmd

{

NETDEV\_PTZ\_ADD\_CRUISE = 0, /\* Add patrol route \*/

NETDEV\_PTZ\_MODIFY\_CRUISE = 1, /\* Edit patrol route \*/

NETDEV\_PTZ\_DEL\_CRUISE = 2, /\* Delete patrol route \*/

NETDEV\_PTZ\_RUN\_CRUISE = 3, /\* Start patrol \*/

NETDEV\_PTZ\_STOP\_CRUISE = 4, /\* Stop patrol \*/

NETDEV\_PTZ\_GET\_CRUISE = 5 /\* Get patrol route \*/

}NETDEV\_PTZ\_CRUISECMD\_E;

### Enumeration of PTZ directions

typedef enum tagNETDEVPTZDirection

{

NETDEV\_PTZ\_DIRECTION\_EAST = 0, /\* East \*/

NETDEV\_PTZ\_DIRECTION\_SOUTHEAST = 1, /\* Southeast \*/

NETDEV\_PTZ\_DIRECTION\_SOUTH = 2, /\* North \*/

NETDEV\_PTZ\_DIRECTION\_SOUTHWEST = 3, /\* Southwest \*/

NETDEV\_PTZ\_DIRECTION\_WEST = 4, /\* West \*/

NETDEV\_PTZ\_DIRECTION\_NORTHWEST = 5, /\* Northwest \*/

NETDEV\_PTZ\_DIRECTION\_NORTH = 6, /\* North \*/

NETDEV\_PTZ\_DIRECTION\_NORTHEAST = 7, /\* Northesat \*/

NETDEV\_PTZ\_DIRECTION\_AUTO = 8, /\* Auto \*/

NETDEV\_PTZ\_DIRECTION\_INVALID = 0xFFFF /\* Invalid value\*/

}NETDEV\_PTZ\_DIRECTION\_E;

### Enumeration of playback control commands

typedef enum tagNETDEVPlayControl

{

NETDEV\_PLAY\_CTRL\_PLAY = 0, /\* Play \*/

NETDEV\_PLAY\_CTRL\_PAUSE = 1, /\* Pause \*/

NETDEV\_PLAY\_CTRL\_RESUME = 2, /\* Resume \*/

NETDEV\_PLAY\_CTRL\_GETPLAYTIME = 3, /\* Get playback progress \*/

NETDEV\_PLAY\_CTRL\_SETPLAYTIME = 4, /\* Set playback progress \*/

NETDEV\_PLAY\_CTRL\_GETPLAYSPEED = 5, /\* Get playback speed \*/

NETDEV\_PLAY\_CTRL\_SETPLAYSPEED = 6, /\* Set playback speed \*/

NETDEV\_PLAY\_CTRL\_SET\_SINGLE\_FRAME\_SPEED = 7, /\* Set playback by frame \*/

NETDEV\_PLAY\_CTRL\_UPDATE\_M3U8URL = 8 /\* Update M3u8URL \*/

}NETDEV\_VOD\_PLAY\_CTRL\_E;

### Enumeration of main log types

typedef enum tagNETDEVLogMainType

{

/\* Supported by NVR only \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALL = 0, /\* All types of logs \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALARM = 1, /\* Alarm \*/

NETDEV\_LOG\_MAIN\_TYPE\_EXCEPTION = 2, /\* Exception \*/

NETDEV\_LOG\_MAIN\_TYPE\_OPERATION = 3, /\* Operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_MESSAGE = 4, /\* Message \*/

/\* VMS operations main log type\*/

NETDEV\_LOG\_MAIN\_TYPE\_VMS\_ALL = 90, /\* All main types \*/

NETDEV\_LOG\_MAIN\_TYPE\_LOGIN = 91, /\* Login \*/

NETDEV\_LOG\_MAIN\_TYPE\_ORGANIZATION\_CONFIG = 100, /\* Organization config \*/

NETDEV\_LOG\_MAIN\_TYPE\_USER\_CONFIG = 101, /\* User config \*/

NETDEV\_LOG\_MAIN\_TYPE\_ROLE\_CONFIG = 102, /\* Role config \*/

NETDEV\_LOG\_MAIN\_TYPE\_DEVICE\_CONFIG = 103, /\* Device config \*/

NETDEV\_LOG\_MAIN\_TYPE\_CHANNE\_LCONFIG = 104, /\* Channel config \*/

NETDEV\_LOG\_MAIN\_TYPE\_SERVER\_CONFIG = 105, /\* primary/secondary server configuration \*/

NETDEV\_LOG\_MAIN\_TYPE\_PTZ\_CONFIG = 106, /\* PTZ config \*/

NETDEV\_LOG\_MAIN\_TYPE\_VIDEOWALL\_CONFIG = 107, /\* Video wall config \*/

NETDEV\_LOG\_MAIN\_TYPE\_EMAP\_CONFIG = 108, /\* Map config \*/

NETDEV\_LOG\_MAIN\_TYPE\_SYSTEM\_CONFIG = 109, /\* System config \*/

NETDEV\_LOG\_MAIN\_TYPE\_SEQUENCE\_PLAN\_CONFIG = 110, /\* Sequence plan config \*/

NETDEV\_LOG\_MAIN\_TYPE\_NETWORK\_KEYPAD\_CONFIG = 111, /\* Network keyboard channel config \*/

NETDEV\_LOG\_MAIN\_TYPE\_SEQUENCE\_RESOURCE\_CONFIG = 112, /\* Sequence resource config \*/

NETDEV\_LOG\_MAIN\_TYPE\_HOT\_SPOT\_AND\_ZONE\_CONFIG = 113, /\* Hot spot & hot zone config \*/

NETDEV\_LOG\_MAIN\_TYPE\_USER\_TIME\_TEMPLATE\_CONFIG = 114, /\* User time template config \*/

NETDEV\_LOG\_MAIN\_TYPE\_SERVER\_WORK\_MODE = 115, /\* Server working mode config \*/

NETDEV\_LOG\_MAIN\_TYPE\_PERMISSION\_TIME\_TEMPLATE\_CONFIG =116, /\* Permission time template config \*/

NETDEV\_LOG\_MAIN\_TYPE\_TIME\_CONFIG = 117, /\* Manual time config \*/

NETDEV\_LOG\_MAIN\_TYPE\_TIMESYNC\_CONFIG = 118, /\* Time sync \*/

NETDEV\_LOG\_MAIN\_TYPE\_HOLIDAY\_CONFIG = 119, /\* Holiday start time \*/

NETDEV\_LOG\_MAIN\_TYPE\_NTPTIME\_CONFIG = 120, /\* NTP time config \*/

NETDEV\_LOG\_MAIN\_TYPE\_DST\_CONFIG = 121, /\* DST time config \*/

NETDEV\_LOG\_MAIN\_TYPE\_RESRELATION\_CONFIG = 122, /\* Related resource config \*/

NETDEV\_LOG\_MAIN\_TYPE\_DEVICE\_TIME\_SYNC = 123, /\* Device time sync \*/

NETDEV\_LOG\_MAIN\_TYPE\_TCPIP\_CONFIG = 124, /\* TCP/IP config \*/

NETDEV\_LOG\_MAIN\_TYPE\_EZCLOUD\_CONFIG = 125, /\* Private cloud config \*/

NETDEV\_LOG\_MAIN\_TYPE\_DDNS\_CONFIG = 126, /\* DDNS config \*/

NETDEV\_LOG\_MAIN\_TYPE\_PORT\_CONFIG = 127, /\* Port config \*/

NETDEV\_LOG\_MAIN\_TYPE\_PORTMAPPING\_CONFIG = 128, /\* Port mapping \*/

NETDEV\_LOG\_MAIN\_TYPE\_EMAIL\_CONFIG = 129, /\* Email config \*/

NETDEV\_LOG\_MAIN\_TYPE\_GBSERVER\_CONFIG = 130, /\* GB server config \*/

NETDEV\_LOG\_MAIN\_TYPE\_GBT28181\_LOCAL\_CONFIG = 131, /\* GB local config \*/

NETDEV\_LOG\_MAIN\_TYPE\_UNP\_CLIENT\_CONFIG = 132, /\* UNP client config \*/

NETDEV\_LOG\_MAIN\_TYPE\_UNP\_SERVER\_CONFIG = 133, /\* UNP server config \*/

NETDEV\_LOG\_MAIN\_TYPE\_STATIC\_ROUTE\_CONFIG = 134, /\* Static route operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_802DOT1X\_CONFIG = 135, /\* 802.1x config \*/

NETDEV\_LOG\_MAIN\_TYPE\_ARP\_CONFIG = 136, /\* ARP config \*/

NETDEV\_LOG\_MAIN\_TYPE\_HTTPS\_CONFIG = 137, /\* HTTPS config \*/

NETDEV\_LOG\_MAIN\_TYPE\_TELNET\_CONFIG = 138, /\* Telnet config \*/

NETDEV\_LOG\_MAIN\_TYPE\_SECURITY\_PSW\_CONFIG = 139, /\* Security password config \*/

NETDEV\_LOG\_MAIN\_TYPE\_IP\_FILTER\_RULE\_CONFIG = 140, /\* IP filtering rule config \*/

NETDEV\_LOG\_MAIN\_TYPE\_MAINTENANCE\_CONFIG = 141, /\* System maintenance \*/

NETDEV\_LOG\_MAIN\_TYPE\_SET\_MAX\_LOG\_RETENTION\_TIME = 142, /\* Log retention time config \*/

NETDEV\_LOG\_MAIN\_TYPE\_STREAM\_TRANSMISSION\_POLICY\_CONFIG = 143, /\* Media stream transmission policy config \*/

NETDEV\_LOG\_MAIN\_TYPE\_BATCH\_CONFIG = 144, /\* Batch config \*/

NETDEV\_LOG\_MAIN\_TYPE\_RAIDMODE\_CONFIG = 145, /\* RAID mode config \*/

NETDEV\_LOG\_MAIN\_TYPE\_ARRAY\_CONFIG = 146, /\* Array config \*/

NETDEV\_LOG\_MAIN\_TYPE\_DISK\_MANAGEMENT\_CONFIG = 147, /\* HDD management \*/

NETDEV\_LOG\_MAIN\_TYPE\_CHANNEL\_SPACE\_CONFIG = 148, /\* Channel space config \*/

NETDEV\_LOG\_MAIN\_TYPE\_OVER\_WRITE\_POLICY\_CONFIG = 149, /\* Overwrite policy config \*/

NETDEV\_LOG\_MAIN\_TYPE\_DISKTEST\_CONFIG = 150, /\* HDD test config \*/

NETDEV\_LOG\_MAIN\_TYPE\_NET\_DISK\_CONFIG = 151, /\* NAS config \*/

NETDEV\_LOG\_MAIN\_TYPE\_RECORDING\_SCHEDULE\_CONFIG = 152, /\* Recording schedule config \*/

NETDEV\_LOG\_MAIN\_TYPE\_RECORDING\_TIME\_TEMPLATE\_CONFIG = 153, /\* Recording time template config \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALARM\_CONFIG = 154, /\* Alarm config \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALARM\_TIMETEMPLATE\_CONFIG = 155, /\* Alarm time template config \*/

NETDEV\_LOG\_MAIN\_TYPE\_CONTACT\_CONFIG = 156, /\* Contact config \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALARM\_TO\_VIDEO\_WALL\_CONFIG = 157, /\* Alarm triggered video to video wall config \*/

NETDEV\_LOG\_MAIN\_TYPE\_VIDEO\_WALL\_SCENE\_CONFIG = 158, /\* Video wall scene config \*/

NETDEV\_LOG\_MAIN\_TYPE\_VIDEO\_WALL\_WINDOW\_CONFIG = 159, /\* Video wall window config \*/

NETDEV\_LOG\_MAIN\_TYPE\_VIDEO\_WALL\_VIRTUALLED\_CONFIG = 160, /\* Video wall virtual LED config \*/

NETDEV\_LOG\_MAIN\_TYPE\_EXTERNAL\_ALARM\_CONFIG = 161, /\* Emergency bell config \*/

NETDEV\_LOG\_MAIN\_TYPE\_DISK\_GROUP\_CONFIG = 162, /\* Disk group config \*/

NETDEV\_LOG\_MAIN\_TYPE\_BK\_PLAN\_CONFIG = 163, /\* Backup schedule config \*/

NETDEV\_LOG\_MAIN\_TYPE\_BK\_TASK\_CONFIG = 164, /\* Backup task \*/

NETDEV\_LOG\_MAIN\_TYPE\_LOCAL\_BK\_TASKCONFIG = 165, /\* Local backup task \*/

NETDEV\_LOG\_MAIN\_TYPE\_TVWALL\_AUDIO\_CONFIG = 166, /\* Video wall audio output config \*/

NETDEV\_LOG\_MAIN\_TYPE\_LIVEVIEW = 167, /\* Live view operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_PLAYBACK = 168, /\* Recording playback operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_TWO\_WAY\_AUDIO = 169, /\* Audio operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_PTZ = 170, /\* PTZ operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_DOWNLOAD = 171, /\* Download operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_LIVEVIEW\_ON\_VIDEO\_WALL = 172, /\* Live view on video wall \*/

NETDEV\_LOG\_MAIN\_TYPE\_PLAYBACK\_ON\_VIDEO\_WALL = 173, /\* Playback on video wall \*/

NETDEV\_LOG\_MAIN\_TYPE\_EMAP\_OPERATION = 174, /\* E-map operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_DC\_SEQUENCE\_OPERATION = 175, /\* DC sequence operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_VIDEO\_WALL\_PLAYING\_BY\_NETWROK\_KEYPAD = 176, /\* Play live video on video wall using network keyboard \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALARM\_TO\_VIDEO\_WALL = 177, /\* Alarm linkage to video wall \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALARM\_SUBSCRIPTION = 178, /\* Alarm subscription \*/

NETDEV\_LOG\_MAIN\_TYPE\_PRESET\_PATROL\_CONFIG = 179, /\* Preset patrol config \*/

NETDEV\_LOG\_MAIN\_TYPE\_AUTOGUARD\_CONFIG = 180, /\* Auto guard config \*/

NETDEV\_LOG\_MAIN\_TYPE\_RECORDED\_PATROL\_CONFIG = 181, /\* Recorded patrol config \*/

NETDEV\_LOG\_MAIN\_TYPE\_ACCESS\_CONTROL\_PERSONNEL\_MANAGEMENT = 182, /\* Access control personnel management \*/

NETDEV\_LOG\_MAIN\_TYPE\_ACCESS\_CONTROL\_CARD\_MANAGEMENT = 183, /\* Access control card management \*/

NETDEV\_LOG\_MAIN\_TYPE\_ZONE\_OPERATION = 184, /\* Alarm zone operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_ACCESS\_CONTROL\_DOOR = 185, /\* Door operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_TRIGGER\_ALARMOUTPUT = 186, /\* Trigger alarm output \*/

NETDEV\_LOG\_MAIN\_TYPE\_CENTER\_RECORD = 187, /\* Central recording \*/

NETDEV\_LOG\_MAIN\_TYPE\_FACE\_LIBRARY\_OPERATION = 188, /\* Face library operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_FACE\_MEMBER\_OPERATION = 189, /\* Face member operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_FACE\_CUSTOM\_OPERATION = 190, /\* Custom face attributes \*/

NETDEV\_LOG\_MAIN\_TYPE\_FACE\_MEMBER\_SORT\_OPERATION = 191, /\* Face member assignment \*/

NETDEV\_LOG\_MAIN\_TYPE\_FACE\_MONITORING\_TASKO\_PERATION = 192, /\* Face monitoring \*/

NETDEV\_LOG\_MAIN\_TYPE\_VEHICLE\_LIBRARY\_OPERATION = 193, /\* Vehicle library operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_VEHICLE\_MEMBER\_SORT\_OPERATION = 194, /\* Vehicle member assignment \*/

NETDEV\_LOG\_MAIN\_TYPE\_VEHICLE\_MEMBER\_OPERATION = 195, /\* Vehicle member operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_CAP\_PAC\_TASK = 196, /\* Capture packets \*/

NETDEV\_LOG\_MAIN\_TYPE\_RECORD\_BK\_CONFIG = 197, /\* Recording backup \*/

NETDEV\_LOG\_MAIN\_TYPE\_USB\_OPERATION = 198, /\* USB flash drive operation \*/

NETDEV\_LOG\_MAIN\_TYPE\_VEHICLE\_MONITORING\_TASK\_OPERATION = 199, /\* Vehicle monitoring \*/

NETDEV\_LOG\_MAIN\_TYPE\_PERMISSION\_ASSIGNMENT\_OPERATION = 200, /\* Permission assignment \*/

NETDEV\_LOG\_MAIN\_TYPE\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_LOG\_MAIN\_TYPE\_E;

### Enumeration of log sub types

typedefenumtagNETDEVLogSubType

{

NETDEV\_LOG\_ALL\_SUB\_TYPES =0x0101, /\* All informational logs \*/

/\* Informational logs \*/

NETDEV\_LOG\_MSG\_HDD\_INFO =300, /\* HDD info \*/

NETDEV\_LOG\_MSG\_SMART\_INFO =301, /\* S.M.A.R.T info \*/

NETDEV\_LOG\_MSG\_REC\_OVERDUE =302, /\* Delete expired recording \*/

NETDEV\_LOG\_MSG\_PIC\_REC\_OVERDUE =303, /\* Delete expired image \*/

/\* Notification logs \*/

NETDEV\_LOG\_NOTICE\_IPC\_ONLINE =310, /\* Device online \*/

NETDEV\_LOG\_NOTICE\_IPC\_OFFLINE =311, /\* Device offline \*/

NETDEV\_LOG\_NOTICE\_ARRAY\_RECOVER =312, /\* Array recovered \*/

NETDEV\_LOG\_NOTICE\_INIT\_ARRARY =313, /\* Array initialized \*/

NETDEV\_LOG\_NOTICE\_REBUILD\_ARRARY =314, /\* Array rebuilt \*/

NETDEV\_LOG\_NOTICE\_POE\_PORT\_STATUS =315, /\* PoE port status \*/

NETDEV\_LOG\_NOTICE\_NETWORK\_PORT\_STATUS =316, /\* Network port status \*/

NETDEV\_LOG\_NOTICE\_DISK\_ONLINE =317, /\* HDD online \*/

/\* Sub log ID of alarm logs \*/

NETDEV\_LOG\_ALARM\_MOTION\_DETECT =350, /\* Motion detection alarm \*/

NETDEV\_LOG\_ALARM\_MOTION\_DETECT\_RESUME =351, /\* Motion detection alarm cleared \*/

NETDEV\_LOG\_ALARM\_VIDEO\_LOST =352, /\* Video loss alarm \*/

NETDEV\_LOG\_ALARM\_VIDEO\_LOST\_RESUME =353, /\* Video loss alarm cleared \*/

NETDEV\_LOG\_ALARM\_VIDEO\_TAMPER\_DETECT =354, /\* Tamper detection alarm \*/

NETDEV\_LOG\_ALARM\_VIDEO\_TAMPER\_RESUME =355, /\* Tamper detection alarm cleared \*/

NETDEV\_LOG\_ALARM\_INPUT\_SW =356, /\* Alarm input (input switch alarm) \*/

NETDEV\_LOG\_ALARM\_INPUT\_SW\_RESUME =357, /\* Alarm input cleared (input switch alarm cleared) \*/

NETDEV\_LOG\_ALARM\_IPC\_ONLINE =358, /\* Device online \*/

NETDEV\_LOG\_ALARM\_IPC\_OFFLINE =359, /\* Device offline \*/

NETDEV\_LOG\_ALARM\_AUDIO\_DETECTION\_START =360, /\* Audio detection started \*/

NETDEV\_LOG\_ALARM\_AUDIO\_DETECTION\_END =361, /\* Audio detection ended \*/

NETDEV\_LOG\_ALARM\_CROSS\_LINE\_DETECT =362, /\* Cross line detection alarm \*/

NETDEV\_LOG\_ALARM\_FACE\_DETECT =363, /\* Face detection alarm \*/

NETDEV\_LOG\_ALARM\_INTRUSION\_DETECT =364, /\* Intrusion detection alarm \*/

NETDEV\_LOG\_ALARM\_POS =365, /\* POS alarm \*/

NETDEV\_LOG\_ALARM\_IMAGETOOBLURRY\_ON =366, /\* Defocus detection started \*/

NETDEV\_LOG\_ALARM\_IMAGETOOBLURRY\_OFF =367, /\* Defocus detection ended \*/

NETDEV\_LOG\_ALARM\_GLOBAL\_SCENE\_CHANGE =368, /\* Scene change detection \*/

NETDEV\_LOG\_ALARM\_AUTO\_TRACK\_ON =369, /\* Auto track started \*/

NETDEV\_LOG\_ALARM\_AUTO\_TRACK\_OFF =370, /\* Auto track ended \*/

NETDEV\_LOG\_ALARM\_CONFLAGRATION\_ON =371, /\* Fire alarm \*/

NETDEV\_LOG\_ALARM\_CONFLAGRATION\_OFF =372, /\* Fire alarm ended \*/

NETDEV\_LOG\_ALARM\_HUMAN\_SHAPE\_DETECT\_ON =373, /\* Human body detection \*/

NETDEV\_LOG\_ALARM\_HUMAN\_SHAPE\_DETECT\_OFF =374, /\* Human body detection ended \*/

NETDEV\_LOG\_ALARM\_ENTERAREA =375, /\* Enter area \*/

NETDEV\_LOG\_ALARM\_LEAVEAREA =376, /\* Leave area \*/

/\* Subtype exception logs \*/

NETDEV\_LOG\_EXCEP\_DISK\_ONLINE =400, /\* HDD online \*/

NETDEV\_LOG\_EXCEP\_DISK\_OFFLINE =401, /\* HDD offline \*/

NETDEV\_LOG\_EXCEP\_DISK\_ERR =402, /\* HDD error \*/

NETDEV\_LOG\_EXCEP\_STOR\_ERR =403, /\* Storage error \*/

NETDEV\_LOG\_EXCEP\_STOR\_ERR\_RECOVER =404, /\* Storage error cleared \*/

NETDEV\_LOG\_EXCEP\_STOR\_DISOBEY\_PLAN =405, /\* Scheduled recording failure \*/

NETDEV\_LOG\_EXCEP\_STOR\_DISOBEY\_PLAN\_RECOVER =406, /\* Schedule recording failure cleared \*/

NETDEV\_LOG\_EXCEP\_ILLEGAL\_ACCESS =407, /\* Illegal access \*/

NETDEV\_LOG\_EXCEP\_IP\_CONFLICT =408, /\* IP address conflict \*/

NETDEV\_LOG\_EXCEP\_NET\_BROKEN =409, /\* Network disconnection \*/

NETDEV\_LOG\_EXCEP\_PIC\_REC\_ERR =410, /\* Snapshot error. Failed to get image file \*/

NETDEV\_LOG\_EXCEP\_VIDEO\_EXCEPTION =411, /\* Video input exception (analog channel only) \*/

NETDEV\_LOG\_EXCEP\_VIDEO\_MISMATCH =412, /\* Video format mismatch \*/

NETDEV\_LOG\_EXCEP\_RESO\_MISMATCH =413, /\* Encoding resolution mismatched with front end \*/

NETDEV\_LOG\_EXCEP\_TEMP\_EXCE =414, /\* Temperature exception \*/

NETDEV\_LOG\_EXCEP\_RUNOUT\_RECORD\_SPACE =415, /\* Running out of recording space \*/

NETDEV\_LOG\_EXCEP\_RUNOUT\_IMAGE\_SPACE =416, /\* Running out of image space \*/

NETDEV\_LOG\_EXCEP\_OUT\_RECORD\_SPACE =417, /\* Recording space used up \*/

NETDEV\_LOG\_EXCEP\_OUT\_IMAGE\_SPACE =418, /\* Image space used up \*/

NETDEV\_LOG\_EXCEP\_ANRIDISASSEMBLY =419, /\* Anti-disassembly alarm \*/

NETDEV\_LOG\_EXCEP\_ANRIDISASSEMBLY\_RECOVER =420, /\* Anti-disassembly alarm cleared \*/

NETDEV\_LOG\_EXCEP\_ARRAY\_DAMAGE =421, /\* Array damaged \*/

NETDEV\_LOG\_EXCEP\_ARRAY\_DEGRADE =422, /\* Array degraded \*/

NETDEV\_LOG\_EXCEP\_RECORD\_SNAPSHOT\_ABNOR =423, /\* Recording/snapshot abnormal \*/

NETDEV\_LOG\_EXCEP\_NET\_BROKEN\_RECOVER =424, /\* Network disconnection cleared \*/

NETDEV\_LOG\_EXCEP\_IP\_CONFLICT\_RECOVER =425, /\* IP address conflict cleared \*/

NETDEV\_LOG\_EXCEP\_DEVICE\_HIGHTEMP =426, /\* Device high temperature alarm \*/

NETDEV\_LOG\_EXCEP\_DEVICE\_LOWTEMP =427, /\* Device low temperature alarm \*/

NETDEV\_LOG\_EXCEP\_DEVICE\_HIGHTEMP\_RECOVER =428, /\* Device high temperature alarm cleared \*/

NETDEV\_LOG\_EXCEP\_DEVICE\_LOWTEMP\_RECOVER =429, /\* Device low temperature alarm cleared \*/

NETDEV\_LOG\_EXCEP\_FAN\_FAULT =430, /\* Device fan fault \*/

NETDEV\_LOG\_EXCEP\_FAN\_FAULT\_RECOVER =431, /\* Device fan fault cleared \*/

NETDEV\_LOG\_EXCEP\_LEDBOX\_HIGHTEMP =432, /\* Device LED box high temperature \*/

NETDEV\_LOG\_EXCEP\_LEDBOX\_HIGHTEMP\_RECOVER =433, /\* Device LED box high temperature cleared \*/

NETDEV\_LOG\_EXCEP\_LEDBOX\_SMOKE =434, /\* Device LED box smoke alarm \*/

NETDEV\_LOG\_EXCEP\_LEDBOX\_SMOKE\_RECOVER =435, /\* Device LED box smoke alarm cleared \*/

/\* Sub logs of operation logs \*/

/\* Services \*/

NETDEV\_LOG\_OPSET\_LOGIN =450, /\* User login \*/

NETDEV\_LOG\_OPSET\_LOGOUT =451, /\* User logout \*/

NETDEV\_LOG\_OPSET\_USER\_ADD =452, /\* Add user \*/

NETDEV\_LOG\_OPSET\_USER\_DEL =453, /\* Delete user \*/

NETDEV\_LOG\_OPSET\_USER\_MODIFY =454, /\* Edit user \*/

NETDEV\_LOG\_OPSET\_START\_REC =455, /\* Start recording \*/

NETDEV\_LOG\_OPSET\_STOP\_REC =456, /\* Stop recording \*/

NETDEV\_LOG\_OPSETR\_PLAY\_DOWNLOAD =457, /\* Playback/download \*/

NETDEV\_LOG\_OPSET\_DOWNLOAD =458, /\* Download \*/

NETDEV\_LOG\_OPSET\_PTZCTRL =459, /\* PTZ control \*/

NETDEV\_LOG\_OPSET\_PREVIEW =460, /\* Live view \*/

NETDEV\_LOG\_OPSET\_REC\_TRACK\_START =461, /\* Start recording patrol route \*/

NETDEV\_LOG\_OPSET\_REC\_TRACK\_STOP =462, /\* Stop recording patrol route \*/

NETDEV\_LOG\_OPSET\_START\_TALKBACK =463, /\* Start two-way audio \*/

NETDEV\_LOG\_OPSET\_STOP\_TALKBACK =464, /\* Stop two-way audio \*/

NETDEV\_LOG\_OPSET\_IPC\_ADD =465, /\* Add IPC \*/

NETDEV\_LOG\_OPSET\_IPC\_DEL =466, /\* Delete IPC \*/

NETDEV\_LOG\_OPSET\_IPC\_SET =467, /\* Set IPC \*/

NETDEV\_LOG\_OPSET\_IPC\_QUICK\_ADD =468, /\* Quick add IPC \*/

NETDEV\_LOG\_OPSET\_IPC\_ADD\_BY\_NETWORK =469, /\* Add IPC by network segment \*/

NETDEV\_LOG\_OPSET\_IPC\_MOD\_IP =470, /\* Edit IPC’s network address \*/

/\* Configuration \*/

NETDEV\_LOG\_OPSET\_DEV\_BAS\_CFG =500, /\* Config device basic info \*/

NETDEV\_LOG\_OPSET\_TIME\_CFG =501, /\* Config device time \*/

NETDEV\_LOG\_OPSET\_SERIAL\_CFG =502, /\* Config device serial port \*/

NETDEV\_LOG\_OPSET\_CHL\_BAS\_CFG =503, /\* Config channel basic info \*/

NETDEV\_LOG\_OPSET\_CHL\_NAME\_CFG =504, /\* Config channel name \*/

NETDEV\_LOG\_OPSET\_CHL\_ENC\_VIDEO =505, /\* Config video encoding parameters \*/

NETDEV\_LOG\_OPSET\_CHL\_DIS\_VIDEO =506, /\* Config channel’s video display parameters \*/

NETDEV\_LOG\_OPSET\_PTZ\_CFG =507, /\* Config PTZ \*/

NETDEV\_LOG\_OPSET\_CRUISE\_CFG =508, /\* Config patrol route \*/

NETDEV\_LOG\_OPSET\_PRESET\_CFG =509, /\* Config preset \*/

NETDEV\_LOG\_OPSET\_VIDPLAN\_CFG =510, /\* Config recording schedule \*/

NETDEV\_LOG\_OPSET\_MOTION\_CFG =511, /\* Config motion detection \*/

NETDEV\_LOG\_OPSET\_VIDLOSS\_CFG =512, /\* Config video loss \*/

NETDEV\_LOG\_OPSET\_COVER\_CFG =513, /\* Config video tampering \*/

NETDEV\_LOG\_OPSET\_MASK\_CFG =514, /\* Config privacy mask \*/

NETDEV\_LOG\_OPSET\_SCREEN\_OSD\_CFG =515, /\* Config OSD \*/

NETDEV\_LOG\_OPSET\_ALARMIN\_CFG =516, /\* Config alarm input \*/

NETDEV\_LOG\_OPSET\_ALARMOUT\_CFG =517, /\* Config alarm output \*/

NETDEV\_LOG\_OPSET\_ALARMOUT\_OPEN\_MAN =518, /\* Manually enable alarm output (local interface) \*/

NETDEV\_LOG\_OPSET\_ALARMOUT\_CLOSE\_MAN =519, /\* Manually disable alarm output (local interface) \*/

NETDEV\_LOG\_OPSET\_ABNORMAL\_CFG =520, /\* Config alert \*/

NETDEV\_LOG\_OPSET\_HDD\_CFG =521, /\* Config HDD \*/

NETDEV\_LOG\_OPSET\_NET\_IP\_CFG =522, /\* Config TCP/IP \*/

NETDEV\_LOG\_OPSET\_NET\_PPPOE\_CFG =523, /\* Config PPPoE \*/

NETDEV\_LOG\_OPSET\_NET\_PORT\_CFG =524, /\* Config port \*/

NETDEV\_LOG\_OPSET\_NET\_DDNS\_CFG =525, /\* Config DDNS \*/

NETDEV\_LOG\_OPSET\_AUDIO\_DETECT =527, /\* Config audio detection \*/

NETDEV\_LOG\_OPSET\_SEARCH\_EX\_DISK =528, /\* Search expansion HDD \*/

NETDEV\_LOG\_OPSET\_ADD\_EX\_DISK =529, /\* Add expansion HDD \*/

NETDEV\_LOG\_OPSET\_DEL\_EX\_DISK =530, /\* Delete expansion HDD \*/

NETDEV\_LOG\_OPSET\_SET\_EX\_DISK =531, /\* Config expansion HDD \*/

NETDEV\_LOG\_OPSET\_LIVE\_BY\_MULTICAST =532, /\* Live view by multicast \*/

NETDEV\_LOG\_OPSET\_BISC\_DEV\_INFO =533, /\* Config device basic info \*/

NETDEV\_LOG\_OPSET\_PREVIEW\_CFG =534, /\* Config local preview \*/

NETDEV\_LOG\_OPSET\_SET\_EMAIL =535, /\* Config email \*/

NETDEV\_LOG\_OPSET\_TEST\_EMAIL =536, /\* Test email \*/

NETDEV\_LOG\_OPSET\_SET\_IPCONTROL =537, /\* Config IP permission \*/

NETDEV\_LOG\_OPSET\_PORT\_MAP =538, /\* Config port mapping \*/

NETDEV\_LOG\_OPSET\_ADD\_TAG =539, /\* Add recording tag \*/

NETDEV\_LOG\_OPSET\_DEL\_TAG =540, /\* Delete recording tag \*/

NETDEV\_LOG\_OPSET\_MOD\_TAG =541, /\* Edit recording tag \*/

NETDEV\_LOG\_OPSET\_LOCK\_RECORD =542, /\* Lock recording \*/

NETDEV\_LOG\_OPSET\_UNLOCK\_RECORD =543, /\* Unlock recording \*/

NETDEV\_LOG\_OPSET\_DDNS\_UPDATE\_SUCCESS =545, /\* DDNS updated \*/

NETDEV\_LOG\_OPSET\_DDNS\_INCORRECT\_ID =546, /\* DDNS update failed, incorrect username or password \*/

NETDEV\_LOG\_OPSET\_DDNS\_DOMAIN\_NAME\_NOT\_EXIST =547, /\* DDNS update failed, domain name does not exist \*/

NETDEV\_LOG\_OPSET\_DDNS\_UPDATE\_FAIL =548, /\* DDNS update failed \*/

NETDEV\_LOG\_OPSET\_HTTP\_CFG =549, /\* Config HTTPS \*/

NETDEV\_LOG\_OPSET\_IP\_OFFLINE\_ALARM\_CFG =550, /\* Config IPC offline alarm \*/

NETDEV\_LOG\_OPSET\_TELNET\_CFG =551, /\* Config Telnet \*/

NETDEV\_LOG\_OPSET\_TEST\_DDNS\_DOMAIN =552, /\* Test DDNS domain name \*/

NETDEV\_LOG\_OPSET\_DDNS\_DOMAIN\_CONFLICT =553, /\* DDNS domain name conflict \*/

NETDEV\_LOG\_OPSET\_DDNS\_DOMAIN\_INVALID =554, /\* DDNS domain name is invalid \*/

NETDEV\_LOG\_OPSET\_DEL\_PRESET =555, /\* Delete preset \*/

NETDEV\_LOG\_OPSET\_PTZ\_3D\_POSITION =556, /\* 3D positioning \*/

NETDEV\_LOG\_OPSET\_SNAPSHOT\_SCHEDULE\_CFG =557, /\* Config snapshot schedule \*/

NETDEV\_LOG\_OPSET\_IMAGE\_UPLOAD\_SCHEDULE\_CFG =558, /\* Config image upload schedule \*/

NETDEV\_LOG\_OPSET\_FTP\_CFG =559, /\* Config FTP server \*/

NETDEV\_LOG\_OPSET\_TEST\_FTP\_SERVER =560, /\* Test FTP server connectivity \*/

NETDEV\_LOG\_OPSET\_START\_MANUAL\_SNAPSHOT =561, /\* Start manual snapshot \*/

NETDEV\_LOG\_OPSET\_CLOSE\_MANUAL\_SNAPSHOT =562, /\* Stop manual snapshot \*/

NETDEV\_LOG\_OPSET\_SNAPSHOT\_CFG =563, /\* Config snapshot parameters \*/

NETDEV\_LOG\_OPSET\_ADD\_HOLIDAY =564, /\* Add holidy \*/

NETDEV\_LOG\_OPSET\_DEL\_HOLIDAY =565, /\* Delete holiday \*/

NETDEV\_LOG\_OPSET\_MOD\_HOLIDAY =566, /\* Edit holiday \*/

NETDEV\_LOG\_OPSET\_ONOFF\_HOLIDAY =567, /\* Enable/disable holiday \*/

NETDEV\_LOG\_OPSET\_ALLOCATE\_SPACE =568, /\* Allocate space \*/

NETDEV\_LOG\_OPSET\_HDD\_FULL\_POLICY\_CFG =569, /\* Configure overwrite policy \*/

NETDEV\_LOG\_OPSET\_AUDIO\_STREAM\_CFG =570, /\* Config audio stream \*/

NETDEV\_LOG\_OPSET\_ARRAY\_PROPERTY\_CFG =571, /\* Config array property \*/

NETDEV\_LOG\_OPSET\_HOT\_SPACE\_DISK\_CFG =572, /\* Config hot spare HDD \*/

NETDEV\_LOG\_OPSET\_CREAT\_ARRAY =573, /\* Create array manually \*/

NETDEV\_LOG\_OPSET\_ONE\_CLICK\_CREAT\_ARRAY =574, /\* One-click to create array \*/

NETDEV\_LOG\_OPSET\_REBUILD\_ARRAY =575, /\* Rebuild array \*/

NETDEV\_LOG\_OPSET\_DEL\_ARRAY =576, /\* Delete array \*/

NETDEV\_LOG\_OPSET\_ENABLE\_RAID =577, /\* Enable RAID mode \*/

NETDEV\_LOG\_OPSET\_DISABLE\_RAID =578, /\* Disable RAID mode \*/

NETDEV\_LOG\_OPSET\_TEST\_SMART =579, /\* S.M.A.R.T test \*/

NETDEV\_LOG\_OPSET\_SMART\_CFG =580, /\* Config S.M.A.R.T. \*/

NETDEV\_LOG\_OPSET\_BAD\_SECTOR\_DETECT =581, /\* Detect bad sectors \*/

NETDEV\_LOG\_OPSET\_AUDIO\_ALARM\_DURATION =582, /\* Config alarm sound duration \*/

NETDEV\_LOG\_OPSET\_CLR\_AUDIO\_ALARM =583, /\* Clear alarm sound \*/

NETDEV\_LOG\_OPSET\_IPC\_TIME\_SYNC\_CFG =584, /\* Config camera time sync \*/

NETDEV\_LOG\_OPSET\_ENABLE\_DISK\_GROUP =585, /\* Enable disk group \*/

NETDEV\_LOG\_OPSET\_DISABLE\_DISK\_GRRUOP =586, /\* Disable disk group \*/

NETDEV\_LOG\_OPSET\_ONVIF\_AUTH\_CFG =587, /\* Config ONVIF authentication \*/

NETDEV\_LOG\_OPSET\_8021X\_CFG =588, /\* Config 802.1X \*/

NETDEV\_LOG\_OPSET\_ARP\_PROTECTION\_CFG =589, /\* Config ARP protection \*/

NETDEV\_LOG\_OPSET\_SMART\_BASIC\_INFO\_CFG =590, /\* Config smart alarm basic info \*/

NETDEV\_LOG\_OPSET\_CROSS\_LINE\_DETECT\_CFG =591, /\* Config cross line detection \*/

NETDEV\_LOG\_OPSET\_INSTRUSION\_DETECT\_CFG =592, /\* Config intrusion detection \*/

NETDEV\_LOG\_OPSET\_PEOPLE\_COUNT\_CFG =593, /\* Config people counting \*/

NETDEV\_LOG\_OPSET\_FACE\_DETECT\_CFG =594, /\* Config face detection \*/

NETDEV\_LOG\_OPSET\_FISHEYE\_CFG =595, /\* Config fisheye \*/

NETDEV\_LOG\_OPSET\_CUSTOM\_PROTOCOL\_CFG =596, /\* Config custom protocol \*/

NETDEV\_LOG\_OPSET\_BEHAVIOR\_SEARCH =597, /\* Behavior search \*/

NETDEV\_LOG\_OPSET\_FACE\_SEARCH =598, /\* Face search \*/

NETDEV\_LOG\_OPSET\_PEOPLE\_COUNT =599, /\* People counting \*/

/\* Maintenance \*/

NETDEV\_LOG\_OPSET\_START\_DVR =600, /\* Start device \*/

NETDEV\_LOG\_OPSET\_STOP\_DVR =601, /\* Stop device \*/

NETDEV\_LOG\_OPSET\_REBOOT\_DVR =602, /\* Restart device \*/

NETDEV\_LOG\_OPSET\_UPGRADE =603, /\* Upgrade version \*/

NETDEV\_LOG\_OPSET\_LOGFILE\_EXPORT =604, /\* Export log file \*/

NETDEV\_LOG\_OPSET\_CFGFILE\_EXPORT =605, /\* Export configuration file \*/

NETDEV\_LOG\_OPSET\_CFGFILE\_IMPORT =606, /\* Import configuration file \*/

NETDEV\_LOG\_OPSET\_CONF\_SIMPLE\_INIT =607, /\* Restore defaults \*/

NETDEV\_LOG\_OPSET\_CONF\_ALL\_INIT =608, /\* Restore factory defaults \*/

NETDEV\_LOG\_OPSET\_VCA\_BACKUP =700, /\* VCA backup \*/

NETDEV\_LOG\_OPSET\_3G4G\_CFG =701, /\* Config 3G/4G \*/

NETDEV\_LOG\_OPSET\_MOUNT\_EXTENDED\_DISK =702, /\* Mount expansion HDD \*/

NETDEV\_LOG\_OPSET\_UNMOUNT\_EXTENDED\_DISK =703, /\* Unmount expansion HDD \*/

NETDEV\_LOG\_OPSET\_FORCE\_USER\_OFFLINE =704, /\* Force user offline \*/

NETDEV\_LOG\_OPSET\_AUTO\_FUNCTION =709, /\* Automatic maintenance \*/

NETDEV\_LOG\_OPSET\_IPC\_UPRAGDE =710, /\* Upgrade camera \*/

NETDEV\_LOG\_OPSET\_RESTORE\_IPC\_DEFAULTS =711, /\* Restore camera defaults \*/

NETDEV\_LOG\_OPSET\_ADD\_TRANSACTION =712, /\* Add transaction \*/

NETDEV\_LOG\_OPSET\_MOD\_TRANSACTION =713, /\* Edit transaction \*/

NETDEV\_LOG\_OPSET\_DEL\_TRANSACTION =714, /\* Delete transaction \*/

NETDEV\_LOG\_OPSET\_POS\_OSD =715, /\* Config POS display \*/

NETDEV\_LOG\_OPSET\_ADD\_HOT\_SPACE\_DEV =716, /\* Add hot spare device \*/

NETDEV\_LOG\_OPSET\_DEL\_HOT\_SPACE\_DEV =717, /\* Delete hot spare device \*/

NETDEV\_LOG\_OPSET\_MOD\_HOT\_SPACE\_DEV =718, /\* Edit hot spare device \*/

NETDEV\_LOG\_OPSET\_DEL\_WORK\_DEV =719, /\* Delete working device \*/

NETDEV\_LOG\_OPSET\_WORKMODE\_TO\_NORMAL\_CFG =720, /\* Set working mode \*/

NETDEV\_LOG\_OPSET\_WORKMODE\_TO\_HOTSPACE\_CFG =721, /\* Set hot spare mode \*/

NETDEV\_LOG\_OPSET\_AUTO\_GUARD\_CFG =723, /\* Config auto guard \*/

NETDEV\_LOG\_OPSET\_MULTICAST\_CFG =724, /\* Config multicast \*/

NETDEV\_LOG\_OPSET\_DEFOCUS\_DETECT\_CFG =725, /\* Config defocus detection \*/

NETDEV\_LOG\_OPSET\_SCENECHANGE\_CFG =726, /\* Config scene change detection \*/

NETDEV\_LOG\_OPSET\_AUTO\_TRCAK\_CFG =727, /\* Config auto track \*/

NETDEV\_LOG\_OPSET\_SORT\_CAMERA\_CFG =728, /\* Sort channels \*/

NETDEV\_LOG\_OPSET\_WATER\_MARK\_CFG =729, /\* Config video watermark \*/

/\* Subtype operation logs of VMS \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_ALL =1000, /\* All sub types \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USER\_LOGIN =1001, /\* User login \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USER\_LOGOUT =1002, /\* User logout \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USER\_START\_OPERATION =1003, /\* User start \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USER\_STOP\_OPERATION =1004, /\* User stop \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_NEW\_CONFIG =1005, /\* Add configuration \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_EDIT\_CONFIG =1006, /\* Edit configuration \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_DELETE\_CONFIG =1007, /\* Delete configuration \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_ENABLE\_CONFIG =1008, /\* Enable configuration \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_DISABLE\_CONFIG =1009, /\* Disable configuration \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_TEST\_CONFIG =1010, /\* Test configuration \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_OPEN\_DOOR =1011, /\* Open door \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_CLOSE\_DOOR =1012, /\* Close door \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_SARM =1013, /\* Arm \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_DISARM =1014, /\* Disarm \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_BYPASS =1015, /\* Bypass \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_UNBYPASS =1016, /\* Unbypass \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_SHARE\_DEVICE =1017, /\* Share cloud device \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_UN\_SHARE\_DEVICE =1018, /\* Cancel sharing \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_START\_CAP\_PAC\_TASK =1019, /\* Start packet capturing task \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_STOP\_CAP\_PAC\_TASK =1020, /\* Stop packet capturing task \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_DELETE\_CAPPAC\_TASK=1021, /\* Delete packet capturing task \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USB\_INSERT =1022, /\* Insert USB disk\*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USB\_PULL\_OUT =1023, /\* Remove USB disk \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USB\_FORMAT =1024, /\* Format USB disk \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_SYNC\_DEV\_CHL\_INFO =1025, /\* Sync device channel info \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_PLAYBACK\_LOCK =1026, /\* Lock recording \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_PLAYBACK\_UNLOCK =1027 /\* Unlock recording \*/

}NETDEV\_LOG\_SUB\_TYPE\_E;

### Enumeration of alarm types

typedef enum tagNETDEVFindAlarmTypeEn

{

NETDEV\_FIND\_ALARM\_DISK\_STORAGE\_IS\_FULL = 1, /\* Storage full \*/

NETDEV\_FIND\_ALARM\_DISK\_STORAGE\_WILL\_FULL = 2, /\* Running out of storage\*/

NETDEV\_FIND\_ALARM\_STOR\_ERR = 3, /\* Storage error \*/

NETDEV\_FIND\_ALARM\_STOR\_ERR\_RECOVER = 4, /\* Storage error cleared \*/

NETDEV\_FIND\_ALARM\_MOVE\_DETECT = 5, /\* Motion detection alarm \*/

NETDEV\_FIND\_ALARM\_MOVE\_DETECT\_RECOVER = 6, /\* Motion detection alarm cleared \*/

NETDEV\_FIND\_ALARM\_VIDEO\_TAMPER\_DETECT = 7, /\* Tampering detection alarm \*/

NETDEV\_FIND\_ALARM\_VIDEO\_TAMPER\_RECOVER = 8, /\* Tampering detection alarm cleared \*/

NETDEV\_FIND\_ALARM\_INPUT\_SWITCH = 9, /\* Alarm input (input switch alarm) \*/

NETDEV\_FIND\_ALARM\_INPUT\_SWITCH\_RECOVER = 10, /\* Alarm input cleared (input switch alarm) \*/

NETDEV\_FIND\_ALARM\_BAND\_CHANGE = 11, /\* Device’s egress bandwidth changed \*/

NETDEV\_FIND\_ALARM\_IMAGE\_BLURRY = 12, /\* Image defocus \*/

NETDEV\_FIND\_ALARM\_IMAGE\_BLURRY\_RECOVER = 13, /\* Blurry image cleared \*/

NETDEV\_FIND\_ALARM\_SCENE\_CHANGE = 14, /\* Scene change \*/

NETDEV\_FIND\_ALARM\_ILLEGAL\_ACCESS = 15, /\* Illegal access \*/

NETDEV\_FIND\_ALARM\_REPORT\_DEV\_ONLINE = 16, /\* Device online \*/

NETDEV\_FIND\_ALARM\_REPORT\_DEV\_OFFLINE = 17, /\* Device offline \*/

NETDEV\_FIND\_ALARM\_REPORT\_DEV\_VIDEO\_LOSS = 18, /\* Video loss \*/

NETDEV\_FIND\_ALARM\_REPORT\_DEV\_VIDEO\_LOSS\_RECOVER = 19, /\* Video loss cleared \*/

NETDEV\_FIND\_ALARM\_COUNT\_PEOPLE = 20, /\* People counting \*/

NETDEV\_FIND\_ALARM\_HEAT\_MAP = 21, /\* Heatmap \*/

NETDEV\_FIND\_ALARM\_LINE\_DETECTOR = 22, /\* Tripwire alarm \*/

NETDEV\_FIND\_ALARM\_CELL\_MOTION = 23, /\* Smart motion detection \*/

NETDEV\_FIND\_ALARM\_LINE\_CROSS = 24, /\* Cross line detection \*/

NETDEV\_FIND\_ALARM\_OBJECTS\_INSIDE = 25, /\* Intrusion detection \*/

NETDEV\_FIND\_ALARM\_ACCESSZONE = 26, /\* Enter area alarm \*/

NETDEV\_FIND\_ALARM\_LEAVE\_ZONE = 27, /\* Leave area alarm \*/

NETDEV\_FIND\_ALARM\_HOVER\_ZONE = 28, /\* Loitering alarm \*/

NETDEV\_FIND\_ALARM\_OVER\_ZONE = 29, /\* Crossing fence alarm \*/

NETDEV\_FIND\_ALARM\_CARE\_ARTICLE = 30, /\* Object watching alarm \*/

NETDEV\_FIND\_ALARM\_REMAIN\_ARTICLE = 31, /\* Object left behind alarm \*/

NETDEV\_FIND\_ALARM\_FACE\_DETECTOR = 32, /\* Face detection \*/

NETDEV\_FIND\_ALARM\_GATHER = 33, /\* People gathering alarm \*/

NETDEV\_FIND\_ALARM\_GATHER\_RECOVER = 34, /\* People gathering alarm cleared \*/

NETDEV\_FIND\_ALARM\_FAST\_MOVE = 35, /\* Quick moving \*/

NETDEV\_FIND\_ALARM\_ILLEGAL\_PARKED = 36, /\* Illegal parking alarm \*/

NETDEV\_FIND\_ALARM\_HUMAN\_SHAPE\_ON = 37, /\* Human body detection alarm \*/

NETDEV\_FIND\_ALARM\_HUMAN\_SHAPE\_OFF = 38, /\* Human body detection alarm cleared \*/

NETDEV\_FIND\_ALARM\_BODY\_TEMPERATURE = 39, /\* Abnormal body temperature alarm \*/

NETDEV\_FIND\_ALARM\_NO\_MASK = 40, /\* No mask alarm \*/

NETDEV\_FIND\_ALARM\_METAL\_DETECT\_ON = 41, /\* Metal detector alarm started \*/

NETDEV\_FIND\_ALARM\_METAL\_DETECT\_OFF = 42, /\* Metal detector alarm ended \*/

NETDEV\_FIND\_ALARM\_CROWD\_DENSITY\_MINOR\_ON = 43, /\* Crowd density minor alarm \*/

NETDEV\_FIND\_ALARM\_CROWD\_DENSITY\_MINOR\_OFF = 44, /\* Crowd density minor alarm cleared \*/

NETDEV\_FIND\_ALARM\_CROWD\_DENSITY\_MAJOR\_ON = 45, /\* Crowd density major alarm \*/

NETDEV\_FIND\_ALARM\_CROWD\_DENSITY\_MAJOR\_OFF = 46, /\* Crowd density major alarm cleared \*/

NETDEV\_FIND\_ALARM\_CROWD\_DENSITY\_CRITICAL\_ON = 47, /\* Crowd density critical alarm \*/

NETDEV\_FIND\_ALARM\_CROWD\_DENSITY\_CRITICAL\_OFF = 48, /\* Crowd density critical alarm cleared \*/

NETDEV\_FIND\_ALARM\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_FIND\_ALARM\_TYPE\_E;

### Enumeration of alarm log search criteria

typedef enum tagNETDEVQueryCondType

{

NETDEV\_QUERYCOND\_USERNAME = 0, /\* Username \*/

NETDEV\_QUERYCOND\_ORGNAME = 1, /\* Organization name \*/

NETDEV\_QUERYCOND\_DEVNAME = 2, /\* Device name \*/

NETDEV\_QUERYCOND\_CHNNAME = 3, /\* Channel name \*/

NETDEV\_QUERYCOND\_TIME = 4, /\* Time \*/

NETDEV\_QUERYCOND\_BUSINESSTYPE = 5, /\* Service type \*/

NETDEV\_QUERYCOND\_OPERATETYPE = 6, /\* Operation type \*/

NETDEV\_QUERYCOND\_OPEROBJECT = 7, /\* Operation object \*/

NETDEV\_QUERYCOND\_ALARMTYPE = 8, /\* Alarm type. See [NETDEV\_ALARM\_TYPE\_E](#_告警类型枚举_1)\*/

NETDEV\_QUERYCOND\_ALARMSRCNAME= 9, /\* Alarm source name \*/

NETDEV\_QUERYCOND\_ALARMLEVEL = 10, /\* Alarm level \*/

NETDEV\_QUERYCOND\_ALARMCHECKED = 11, /\* Whether alarm is acknowledged \*/

NETDEV\_QUERYCOND\_ALARMCHECKUSER = 12, /\* User who acknowledged the alarm \*/

NETDEV\_QUERYCOND\_ALARMCHECKTIME = 13, /\* Time when alarm was acknowledged \*/

NETDEV\_QUERYCOND\_ALARM\_DEVID = 14, /\* Alarm device ID \*/

NETDEV\_QUERYCOND\_ALARM\_CHNID = 15, /\* Alarm channel ID \*/

NETDEV\_QUERYCOND\_ALARM\_SUBTYPE = 16, /\* Alarm sub type \*/

NETDEV\_QUERYCOND\_ALARM\_SERVER = 17, /\* Server \*/

NETDEV\_QUERYCOND\_DOOR\_NUM = 18, /\* Door No. \*/

NETDEV\_QUERYCOND\_CARD\_NUM = 19, /\* Card number \*/

NETDEV\_QUERYCOND\_ALARM\_GENDER = 20, /\* Gender \*/

NETDEV\_QUERYCOND\_ALARM\_BIRTHDAY = 21, /\* Date of birth \*/

NETDEV\_QUERYCOND\_MONITOY\_REASON = 22, /\* Monitoring cause \*/

NETDEV\_QUERYCOND\_PLATE\_NUM = 23, /\* Plate number \*/

NETDEV\_QUERYCOND\_VEHICLE\_TYPE = 24, /\* Vehicle type \*/

NETDEV\_QUERYCOND\_PLATE\_COLOR = 25, /\* Plate color \*/

NETDEV\_QUERYCOND\_VEHICLE\_COLOR = 26, /\* Vehicle color \*/

NETDEV\_QUERYCOND\_PERSON\_NUMBER = 27, /\* Person No. \*/

NETDEV\_QUERYCOND\_PERSON\_TYPE = 28, /\* Person type \*/

NETDEV\_QUERYCOND\_DIRECT = 29, /\* Direction \*/

NETDEV\_QUERYCOND\_ORGID = 30, /\* Organization ID \*/

NETDEV\_QUERYCOND\_ORGPID = 31, /\* Organization PID \*/

NETDEV\_QUERYCOND\_DEVICEID = 32, /\* Device ID \*/

NETDEV\_QUERYCOND\_DEVICE\_TYPE = 33, /\* Device type \*/

NETDEV\_QUERYCOND\_DEVICE\_SUBTYPE = 34, /\* Device sub type \*/

NETDEV\_QUERYCOND\_CHANNELID = 35, /\* Channel ID \*/

NETDEV\_QUERYCOND\_CHANNEL\_TYPE = 36, /\* Channel type \*/

NETDEV\_QUERYCOND\_ONLINE\_STATE = 37, /\* online status \*/

NETDEV\_DATABASE\_ID = 38, /\* Library ID \*/

NETDEV\_QUERY\_TYPE\_PLATECLASS = 39, /\* Plate type \*/

NETDEV\_QUERYCOND\_RANGE = 40, /\* Alarm search range 0: Device 1: Server \*/

NETDEV\_QUERYCOND\_BEGIN\_TIME = 41, /\* Scheduled start time of visit \*/

NETDEV\_QUERYCOND\_END\_TIME = 42, /\* Schedule end time of visit \*/

NETDEV\_QUERYCOND\_INTERVIEWEE\_NAME = 43, /\* Name of the visited \*/

NETDEV\_QUERYCOND\_INTERVIEWEE\_STATUS= 44, /\* Status of the visited \*/

NETDEV\_QUERYCOND\_PARK\_NAME = 45, /\* Parking lot name \*/

NETDEV\_QUERYCOND\_CONFIDENCE\_LEVEL = 46, /\* Confidence level \*/

NETDEV\_QUERYCOND\_PARK\_TIME = 47, /\* Parking duration \*/

NETDEV\_QUERYCOND\_CONTRACT\_RULE = 48, /\* Contract rules \*/

NETDEV\_QUERYCOND\_PAYMENT\_METHOD = 49, /\* Payment method \*/

NETDEV\_QUERYCOND\_PASSING\_DIRECTION= 50, /\* Passing direction \*/

NETDEV\_QUERYCOND\_VEHICLE\_ATTR = 51, /\* Vehicle attribute \*/

NETDEV\_QUERYCOND\_STATISTICS\_UNITS = 52, /\* Statistics unit \*/

NETDEV\_QUERYCOND\_EXITENTRANCE\_NAME= 53, /\* Entrance & exit name \*/

NETDEV\_QUERYCOND\_PICTURE\_DATA = 54, /\* Image data \*/

NETDEV\_QUERYCOND\_PERSON\_NAME = 55, /\* Person name \*/

NETDEV\_QUERYCOND\_SIMILARITY = 56, /\* Similarity \*/

NETDEV\_QUERYCOND\_SEARCH\_TYPE = 57, /\* Search type. See [NETDEV\_SEARCH\_TYPE\_E](#_人脸查找类型) \*/

NETDEV\_QUERYCOND\_ID\_NUMBER = 58, /\* ID No. \*/

NETDEV\_QUERYCOND\_AGERANGE = 59, /\* Age range \*/

NETDEV\_QUERYCOND\_GLASSES\_STYLE = 61, /\* Glasses style \*/

NETDEV\_QUERYCOND\_SLEEVES\_LENGTH = 62, /\* Sleeve length \*/

NETDEV\_QUERYCOND\_COAT\_COLOR = 63, /\* Coat color \*/

NETDEV\_QUERYCOND\_TROUSERS\_STYLE = 64, /\* Trousers length \*/

NETDEV\_QUERYCOND\_TROUSERS\_COLOR = 65, /\* Trousers color \*/

NETDEV\_QUERYCOND\_SNAP\_TOWARD = 66, /\* Snapshot direction \*/

NETDEV\_QUERYCOND\_SHOES\_TUBE\_LENGTH= 67, /\* Shoes tube length \*/

NETDEV\_QUERYCOND\_HAIR\_LENGTH = 68, /\* Hair length \*/

NETDEV\_QUERYCOND\_BAG\_FLAG = 69, /\* With or without bag \*/

NETDEV\_QUERYCOND\_SPEED\_TYPE = 70, /\* Speed type \*/

NETDEV\_QUERYCOND\_NON\_VEH\_TYPE = 71, /\* Non-motor vehicle type \*/

NETDEV\_QUERYCOND\_VEH\_BRAND = 72, /\* Vehicle brand \*/

NETDEV\_QUERYCOND\_VEH\_DATA\_TYPE = 73, /\* Vehicle data type (0: normal snapshot 1: structured snapshot) \*/

NETDEV\_QUERYCOND\_PROTOCOL\_TYPE = 74, /\* Device connection protocol \*/

NETDEV\_QUERYCOND\_RELEVANT\_ROOM = 75, /\* Related room \*/

NETDEV\_QUERYCOND\_LOCK\_SIGNAL = 76, /\* Smart lock signal \*/

NETDEV\_QUERYCOND\_BIND\_RELATION\_DOORLOCK = 77, /\* Door-lock binding relationship \*/

NETDEV\_QUERYCOND\_BIND\_RELATION\_PERSON\_CARD = 78, /\* Person-card binding relationship \*/

NETDEV\_QUERYCOND\_PERSONID = 79, /\* Person ID \*/

NETDEV\_QUERYCOND\_PARKINGLOTID = 80, /\* Parking lot ID \*/

NETDEV\_QUERYCOND\_ENTREXITID = 81, /\* Entrance/exit ID \*/

NETDEV\_QUERYCOND\_RECORDID = 82, /\* Record ID \*/

NETDEV\_QUERYCOND\_VEH\_GROUPINGID = 83, /\* Vehicle group ID \*/

NETDEV\_QUERYCOND\_KEYWORD = 84, /\* Keyword \*/

NETDEV\_QUERYCOND\_PUBULISH\_TYPE = 85, /\* Publish type (0: instant publish 1: scheduled publish) \*/

NETDEV\_QUERYCOND\_PROGRAM\_NAME = 86, /\* Program name \*/

NETDEV\_QUERYCOND\_ORG\_TYPE = 87, /\* Organization type. See [NETDEV\_ORG\_TYPE\_E](#_组织类型枚举) \*/

NETDEV\_QUERYCOND\_VALID\_DATE = 88, /\* Valid date \*/

NETDEV\_QUERYCOND\_VALID\_TIME = 89, /\* Valid time \*/

NETDEV\_QUERYCOND\_VALID\_WEEK = 90, /\* Valid week \*/

NETDEV\_QUERYCOND\_MODEL\_STATUS\_TYPE= 91,

/\* Face modeling status. See [NETDEV\_MODEL\_STATUS\_E](#_建模状态枚举) \*/

NETDEV\_QUERYCOND\_MASK\_FLAG = 92, /\* With or without mask. See [NETDEV\_MASK\_FLAG\_E](#_是否戴口罩枚举) \*/

NETDEV\_QUERYCOND\_TEMPERATURE = 93, /\* Body temperature \*/

NETDEV\_QUERYCOND\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_QUERYCOND\_TYPE\_E;

### Enumeration of alarm types

typedef enum tagNETDEVAlarmTypeEn

{

NETDEV\_ALARM\_MOVE\_DETECT = 1, /\* Motion detection alarm \*/

NETDEV\_ALARM\_MOVE\_DETECT\_RECOVER = 2, /\* Motion detection alarm cleared \*/

NETDEV\_ALARM\_VIDEO\_LOST = 3, /\* Video loss alarm \*/

NETDEV\_ALARM\_VIDEO\_LOST\_RECOVER = 4, /\* Video loss alarm cleared \*/

NETDEV\_ALARM\_VIDEO\_TAMPER\_DETECT = 5, /\* Tampering detection alarm \*/

NETDEV\_ALARM\_VIDEO\_TAMPER\_RECOVER = 6, /\* Tampering detection alarm cleared \*/

NETDEV\_ALARM\_INPUT\_SWITCH = 7, /\* Alarm input (input switch alarm)\*/

NETDEV\_ALARM\_INPUT\_SWITCH\_RECOVER = 8, /\* Alarm input cleared (input switch alarm cleared) \*/

NETDEV\_ALARM\_TEMPERATURE\_HIGH = 9, /\* High temperature alarm \*/

NETDEV\_ALARM\_TEMPERATURE\_LOW = 10, /\* Low temperature alarm \*/

NETDEV\_ALARM\_TEMPERATURE\_RECOVER = 11, /\* High temperature alarm cleared \*/

NETDEV\_ALARM\_AUDIO\_DETECT = 12, /\* Audio detection alarm \*/

NETDEV\_ALARM\_AUDIO\_DETECT\_RECOVER = 13, /\* Audio detection alarm cleared \*/

NETDEV\_ALARM\_SERVER\_FAULT = 18, /\* Server fault \*/

NETDEV\_ALARM\_SERVER\_NORMAL = 19, /\* Server fault cleared \*/

NETDEV\_ALARM\_REPORT\_DEV\_ONLINE = 201, /\* Device online alarm \*/

NETDEV\_ALARM\_REPORT\_DEV\_OFFLINE = 202, /\* Device offline alarm \*/

NETDEV\_ALARM\_REPORT\_DEV\_REBOOT = 203, /\* Device restart \*/

NETDEV\_ALARM\_REPORT\_DEV\_SERVICE\_REBOOT = 204, /\* Service restart \*/

NETDEV\_ALARM\_REPORT\_DEV\_CHL\_ONLINE = 205, /\* Video channel: Online \*/

NETDEV\_ALARM\_REPORT\_DEV\_CHL\_OFFLINE = 206, /\* Video channel: Offline \*/

NETDEV\_ALARM\_REPORT\_DEV\_DELETE\_CHL = 207, /\* Video channel: Delete \*/

NETDEV\_ALARM\_DEVICE\_HIGHTEMP = 246, /\* Alert: Device high temperature \*/

NETDEV\_ALARM\_DEVICE\_LOWTEMP = 247, /\* Alert: Device low temperature \*/

NETDEV\_ALARM\_FAN\_FAULT = 248, /\* Alert: Fan fault \*/

NETDEV\_ALARM\_LEDBOX\_HIGHTEMP = 249, /\* Alert: LED box high temperature \*/

NETDEV\_ALARM\_LEDBOX\_SMOKE = 250, /\* Alert: LED box smoke alarm \*/

NETDEV\_ALARM\_DEVICE\_HIGHTEMP\_RECOVER = 251, /\* Alert: Device high temperature cleared \*/

NETDEV\_ALARM\_DEVICE\_LOWTEMP\_RECOVER = 252, /\* Alert: Device low temperature cleared \*/

NETDEV\_ALARM\_FAN\_FAULT\_RECOVER = 253, /\* Alert: Fan fault cleared \*/

NETDEV\_ALARM\_LEDBOX\_HIGHTEMP\_RECOVER = 254, /\* Alert: LED box high temperature cleared \*/

NETDEV\_ALARM\_LEDBOX\_SMOKE\_RECOVER = 255, /\* Alert: LED box smoke alarm cleared \*/

NETDEV\_ALARM\_THERM\_HIGH\_TEMPERATURE = 256, /\* High temperature alarm \*/

NETDEV\_ALARM\_THERM\_LOW\_TEMPERATURE = 257, /\* Low temperature alarm \*/

NETDEV\_ALARM\_THERM\_ABNORMAL\_TEMPERATURE= 258, /\* Abnormal temperature alarm \*/

/\* Player alarms \*/

NETDEV\_ALARM\_NET\_FAILED = 401, /\* Session network failure \*/

NETDEV\_ALARM\_NET\_TIMEOUT = 402, /\* Session network timeout \*/

NETDEV\_ALARM\_SHAKE\_FAILED = 403, /\* Session interaction error \*/

NETDEV\_ALARM\_STREAMNUM\_FULL = 404, /\* Stream number reached upper limit \*/

NETDEV\_ALARM\_STREAM\_THIRDSTOP = 405, /\* Stream stopped by third party \*/

NETDEV\_ALARM\_FILE\_END = 406, /\* File end \*/

NETDEV\_ALARM\_RTMP\_CONNECT\_FAIL = 407, /\* RTMP connection failure \*/

NETDEV\_ALARM\_RTMP\_INIT\_FAIL = 408, /\* RTMP initialization failure \*/

NETDEV\_ALARM\_STREAM\_DOWNLOAD\_OVER = 409, /\* VMS GB stream download completed \*/

NETDEV\_ALARM\_PLAYBACK\_FINISH = 410, /\* Playback finished \*/

NETDEV\_ALARM\_VIDEO\_RECORD\_PART = 411, /\* Recording segmentation \*/

NETDEV\_ALARM\_FISHEYE\_STREAM\_EXIST = 412, /\* Fisheye stream exists, for reporting only \*/

NETDEV\_ALARM\_FISHEYE\_STREAM\_NOT\_EXIST = 413, /\* Fisheye stream does not exist, for reporting only \*/

NETDEV\_ALARM\_PTZ\_RESOUCE\_FAIL = 414, /\* 4-sensor panoramic PTZ camera resource error \*/

NETDEV\_ALARM\_PTZ\_STREAM\_EXIST = 415, /\* 4-sensor panoramic PTZ camera stream exists, for reporting only \*/

NETDEV\_ALARM\_STREAM\_NOT\_EXIST = 416, /\* 4-sensor panoramic PTZ camera stream does not exist, for reporting only \*/

NETDEV\_ALARM\_INNER\_TIMEOUT = 417, /\* Internal processing timed out \*/

NETDEV\_ALARM\_STREAM\_NOT\_READY = 418, /\* Stream not ready \*/

NETDEV\_ALARM\_KEEP\_ALIVE\_FAILED = 419, /\* Keep-alive failed \*/

NETDEV\_ALARM\_OVER\_ABILITY = 420, /\* Insufficient playback capability \*/

NETDEV\_ALARM\_UNAUTHORIZED = 421, /\* Unauthorized \*/

NETDEV\_ALARM\_FORIBIDDEN = 422, /\* Forbidden \*/

NETDEV\_ALARM\_METHOD\_NOT\_ALLOWED = 423, /\* Method not allowed \*/

NETDEV\_ALARM\_PRECONDITION\_FAILED = 424, /\* Pre-processing failed \*/

NETDEV\_ALARM\_SESSION\_NOT\_FOUND = 425, /\* Session not found \*/

NETDEV\_ALARM\_NOT\_ENOUGH\_BANDWIDTH2 = 426, /\* Bandwidth not enough (RTSP) \*/

NETDEV\_ALARM\_REALPLAY\_ESTABLISHED = 427, /\* Live view already established \*/

NETDEV\_ALARM\_REALPLAY\_RES\_BUSY = 428, /\* Live view display resource is busy \*/

NETDEV\_ALARM\_MULTICAST\_DISABLED = 429, /\* Multicast disabled \*/

NETDEV\_ALARM\_MULTICAST\_PORT\_OCCUPIED = 430, /\* Multicast port occupied \*/

NETDEV\_ALARM\_MULTICAST\_PORT\_EXHAUSTED = 431, /\* Multicast ports exhausted \*/

NETDEV\_ALARM\_MULTICAST\_USER\_NOT\_EXIST = 432, /\* Multicast user does not exist \*/

NETDEV\_ALARM\_CHANNEL\_NOT\_ONLINE = 433 /\* Channel offline \*/

NETDEV\_ALARM\_TALKBACK\_ENCODED\_INVALID = 434, /\* Two-way audio resource code is invalid \*/

NETDEV\_ALARM\_VOICE\_RES\_USED\_BY\_TALKBACK = 435, /\* Audio resource is being used by two-way audio \*/

NETDEV\_ALARM\_TALKBACK\_EXISTS = 436, /\* Two-way audio already exists \*/

NETDEV\_ALARM\_VOICE\_WORK\_NOT\_EXIST = 437, /\* Audio service does not exist \*/

NETDEV\_ALARM\_TALKBACK\_TIMEOUT = 438, /\* Two-way audio timed out \*/

NETDEV\_ALARM\_TALKBACK\_ERROR = 439, /\* Two-way audio failed \*/

NETDEV\_ALARM\_INNER\_ERROR = 440, /\* Device’s internal processing error \*/

NETDEV\_ALARM\_BAD\_REQUEST = 441, /\* Bad request \*/

NETDEV\_ALARM\_PAYMENT\_REQUIRED = 442, /\* Payment required \*/

NETDEV\_ALARM\_NOT\_FOUND = 443, /\* Not found \*/

NETDEV\_ALARM\_NOT\_ACCEPTABLE = 444, /\* Unacceptable \*/

NETDEV\_ALARM\_PROXY\_REQUIRED = 445, /\* Proxy requires authentication \*/

NETDEV\_ALARM\_REQUEST\_TIMEOUT = 446, /\* Request timed out \*/

NETDEV\_ALARM\_GONE = 447, /\* Not on server \*/

NETDEV\_ALARM\_LENGTH\_REQUIRED = 448, /\* Length required \*/

NETDEV\_ALARM\_ENTITY\_TOO\_LARGE = 449, /\* Entity too long \*/

NETDEV\_ALARM\_URI\_TOO\_LARGE = 450, /\* URI too long \*/

NETDEV\_ALARM\_UNSUPPORTED\_TYPE = 451, /\* Unsupported media type \*/

NETDEV\_ALARM\_NOT\_UNDERSTOOD = 452, /\* Parameter not understood \*/

NETDEV\_ALARM\_CONFERENCE\_NOT\_FOUND = 453, /\* Conference not found \*/

NETDEV\_ALARM\_NOT\_ENOUGH\_BANDWIDTH = 454, /\* Bandwidth not enough (RTSP) \*/

NETDEV\_ALARM\_METHOD\_NOT\_VALID = 455, /\* Invalid method in this status \*/

NETDEV\_ALARM\_HEADER\_NOT\_VALID = 456, /\* Invalid header for the resource \*/

NETDEV\_ALARM\_INVALID\_RANGE = 457, /\* Invalid range \*/

NETDEV\_ALARM\_PARAMETER\_READ\_ONLY = 458, /\* Parameter is read only \*/

NETDEV\_ALARM\_AO\_NOT\_ALLOWED = 459, /\* AO is not allowed \*/

NETDEV\_ALARM\_ONLY\_AO\_ALLOWED = 460, /\* Only AO is allowed \*/

NETDEV\_ALARM\_UNSUPPORTED\_TRANSPORT = 461, /\* Unsupported transport mode \*/

NETDEV\_ALARM\_DESTINATION\_UNREACHABLE = 462, /\* Destination is unreachable \*/

NETDEV\_ALARM\_INTERNAL\_SERVER\_ERROR = 463, /\* Server internal error \*/

NETDEV\_ALARM\_NOT\_IMPLEMENTED = 464, /\* Not implemented \*/

NETDEV\_ALARM\_BAD\_GATEWAY = 465, /\* Gateway error \*/

NETDEV\_ALARM\_SERVICE\_UNAVAILABLE = 466, /\* Service unavailable \*/

NETDEV\_ALARM\_GATEWAY\_TIMEOUT = 467, /\* Gateway timed out \*/

NETDEV\_ALARM\_VERSION\_NOT\_SUPPORTED = 468, /\* Unsupported RTSP version \*/

NETDEV\_ALARM\_OPTION\_NOT\_SUPPORTED = 469, /\* Unsupported option \*/

NETDEV\_ALARM\_SESSION\_NOT\_EXIST = 470, /\* Session does not exist \*/

NETDEV\_ALARM\_UNDEFINED\_ERROR = 471, /\* Undefined error \*/

/\* Alarm module \*/

NETDEV\_ALARM\_DISK\_ERROR = 601, /\* Device disk error \*/

NETDEV\_ALARM\_SYS\_DISK\_ERROR = 602, /\* System disk error \*/

NETDEV\_ALARM\_DISK\_ONLINE = 603, /\* Device disk online \*/

NETDEV\_ALARM\_SYS\_DISK\_ONLINE = 604, /\* System disk online \*/

NETDEV\_ALARM\_DISK\_OFFLINE = 605, /\* Device disk offline \*/

NETDEV\_ALARM\_SYS\_DISK\_OFFLINE = 606, /\* System disk offline \*/

NETDEV\_ALARM\_DISK\_ABNORMAL = 607, /\* Disk abnormal \*/

NETDEV\_ALARM\_DISK\_ABNORMAL\_RECOVER = 608, /\* Disk abnormal cleared \*/

NETDEV\_ALARM\_DISK\_STORAGE\_WILL\_FULL = 609, /\* Running out of disk space \*/

NETDEV\_ALARM\_DISK\_STORAGE\_WILL\_FULL\_RECOVER = 610, /\* Running out of disk space cleared \*/

NETDEV\_ALARM\_DISK\_STORAGE\_IS\_FULL = 611, /\* Device storage space is full \*/

NETDEV\_ALARM\_SYS\_DISK\_STORAGE\_IS\_FULL = 612, /\* System disk space is full \*/

NETDEV\_ALARM\_DISK\_STORAGE\_IS\_FULL\_RECOVER= 613, /\* Device storage space is full cleared \*/

NETDEV\_ALARM\_DISK\_RAID\_DISABLED\_RECOVER = 614, /\* RAID damaged cleared \*/

NETDEV\_ALARM\_DISK\_RAID\_DEGRADED = 615, /\* Device RAID degraded \*/

NETDEV\_ALARM\_SYS\_DISK\_RAID\_DEGRADED = 616, /\* System RAID degraded \*/

NETDEV\_ALARM\_DISK\_RAID\_DISABLED = 617, /\* Device RAID damaged \*/

NETDEV\_ALARM\_SYS\_DISK\_RAID\_DISABLED = 618, /\* System RAID damaged \*/

NETDEV\_ALARM\_DISK\_RAID\_DEGRADED\_RECOVER = 619, /\* RAID degraded cleared \*/

NETDEV\_ALARM\_STOR\_GO\_FULL = 620, /\* Device running out of storage alarm \*/

NETDEV\_ALARM\_SYS\_STOR\_GO\_FULL = 621, /\* System running out of storage alarm \*/

NETDEV\_ALARM\_ARRAY\_NORMAL = 622, /\* Device array is normal \*/

NETDEV\_ALARM\_SYS\_ARRAY\_NORMAL = 623, /\* System array is normal \*/

NETDEV\_ALARM\_DISK\_RAID\_RECOVERED = 624, /\* RAID recovered \*/

NETDEV\_ALARM\_STOR\_ERR = 625, /\* Device storage error \*/

NETDEV\_ALARM\_SYS\_STOR\_ERR = 626, /\* System storage error \*/

NETDEV\_ALARM\_STOR\_ERR\_RECOVER = 627, /\* Storage error cleared \*/

NETDEV\_ALARM\_STOR\_DISOBEY\_PLAN = 628, /\* Scheduled recording failure \*/

NETDEV\_ALARM\_STOR\_DISOBEY\_PLAN\_RECOVER = 629, /\* Scheduled recording failure cleared \*/

NETDEV\_ALARM\_BANDWITH\_CHANGE = 801, /\* Device’s egress bandwidth changed \*/

NETDEV\_ALARM\_VIDEOENCODER\_CHANGE = 802, /\* Device’s stream configuration changed alarm \*/

NETDEV\_ALARM\_IP\_CONFLICT = 803, /\* IP conflict alarm \*/

NETDEV\_ALARM\_IP\_CONFLICT\_CLEARED = 804, /\* IP conflict alarm cleared \*/

NETDEV\_ALARM\_NET\_OFF = 805, /\* Network disconnection alarm \*/

NETDEV\_ALARM\_NET\_RESUME\_ON = 806, /\* Network disconnection alarm cleared \*/

NETDEV\_ALRAM\_CONFLAG\_DETECT = 920, /\* Fire detection alarm \*/

NETDEV\_ALARM\_NO\_MASK = 921, /\* No mask \*/

NETDEV\_ALARM\_BODY\_TEMPERATURE = 922, /\* Body temperature \*/

NETDEV\_ALARM\_AREA\_PEOPLE\_COUNT = 923, /\* Area people counting alarm \*/

NETDEV\_ALARM\_AREA\_PEOPLE\_COUNT\_RECOVER = 924, /\* Area people counting alarm cleared \*/

NETDEV\_ALARM\_ILLEGAL\_ACCESS = 1001, /\* Device illegal access \*/

NETDEV\_ALARM\_SYS\_ILLEGAL\_ACCESS = 1002, /\* System illegal access \*/

NETDEV\_ALARM\_LINE\_CROSS = 1003, /\* Cross line alarm \*/

NETDEV\_ALARM\_OBJECTS\_INSIDE = 1004, /\* Intrusion detection alarm \*/

NETDEV\_ALARM\_FACE\_RECOGNIZE = 1005, /\* Face recognition \*/

NETDEV\_ALARM\_IMAGE\_BLURRY = 1006, /\* Defocus alarm\*/

NETDEV\_ALARM\_SCENE\_CHANGE = 1007, /\* Scene changed \*/

NETDEV\_ALARM\_SMART\_TRACK = 1008, /\* Auto track \*/

NETDEV\_ALARM\_LOITERING\_DETECTOR = 1009, /\* Loitering detection \*/

NETDEV\_ALARM\_BANDWIDTH\_CHANGE = 1010, /\* Bandwidth changed \*/

NETDEV\_ALARM\_ALLTIME\_FLAG\_END = 1011, /\* Flag denoting the end of all-day alarms \*/

NETDEV\_ALARM\_MEDIA\_CONFIG\_CHANGE = 1012, /\* Encoding parameters changed \*/

NETDEV\_ALARM\_REMAIN\_ARTICLE = 1013, /\* Object left behind alarm \*/

NETDEV\_ALARM\_PEOPLE\_GATHER = 1014, /\* People gathering alarm \*/

NETDEV\_ALARM\_ENTER\_AREA = 1015, /\* Enter area \*/

NETDEV\_ALARM\_LEAVE\_AREA = 1016, /\* Leave area \*/

NETDEV\_ALARM\_ARTICLE\_MOVE = 1017, /\* Object removed \*/

NETDEV\_ALARM\_SMART\_FACE\_MATCH\_LIST = 1018, /\* Face recognition blocklist alarm \*/

NETDEV\_ALARM\_SMART\_FACE\_MATCH\_LIST\_RECOVER = 1019, /\* Face recognition blocklist alarm cleared \*/

NETDEV\_ALARM\_SMART\_FACE\_MISMATCH\_LIST = 1020, /\* Face not match alarm \*/

NETDEV\_ALARM\_SMART\_FACE\_MISMATCH\_LIST\_RECOVER = 1021, /\* Face not match alarm cleared \*/

NETDEV\_ALARM\_SMART\_VEHICLE\_MATCH\_LIST = 1022, /\* Vehicle match alarm \*/

NETDEV\_ALARM\_SMART\_VEHICLE\_MATCH\_LIST\_RECOVER = 1023, /\* Vehicle match alarm cleared \*/

NETDEV\_ALARM\_SMART\_VEHICLE\_MISMATCH\_LIST = 1024, /\* Vehicle not match alarm \*/

NETDEV\_ALARM\_SMART\_VEHICLE\_MISMATCH\_LIST\_RECOVER= 1025, /\* Vehicle not match alarm cleared \*/

NETDEV\_ALARM\_IMAGE\_BLURRY\_RECOVER = 1026, /\* Defocus alarm cleared \*/

NETDEV\_ALARM\_SMART\_TRACK\_RECOVER = 1027, /\* Auto track alarm cleared \*/

NETDEV\_ALARM\_SMART\_READ\_ERROR\_RATE = 1028, /\* Data reading error rate \*/

NETDEV\_ALARM\_SMART\_SPIN\_UP\_TIME = 1029, /\* Spin up time \*/

NETDEV\_ALARM\_SMART\_START\_STOP\_COUNT = 1030, /\* Number of start/stop \*/

NETDEV\_ALARM\_SMART\_REALLOCATED\_SECTOR\_COUNT = 1031, /\* Number of reallocated sectors \*/

NETDEV\_ALARM\_SMART\_SEEK\_ERROR\_RATE = 1032, /\* Seek error rate \*/

NETDEV\_ALARM\_SMART\_POWER\_ON\_HOURS = 1033, /\* Power on hours (total power-on duration after delivery) The average life of a disk is 30,000 hours \*/

NETDEV\_ALARM\_SMART\_SPIN\_RETRY\_COUNT = 1034, /\* Number of spin retries \*/

NETDEV\_ALARM\_SMART\_CALIBRATION\_RETRY\_COUNT = 1035, /\* Number of calibration retries \*/

NETDEV\_ALARM\_SMART\_POWER\_CYCLE\_COUNT = 1036, /\* Number of power cycles \*/

NETDEV\_ALARM\_SMART\_POWEROFF\_RETRACT\_COUNT = 1037, /\* Number of power-off retracts \*/

NETDEV\_ALARM\_SMART\_LOAD\_CYCLE\_COUNT = 1038, /\* Number of load cycles \*/

NETDEV\_ALARM\_SMART\_TEMPERATURE\_CELSIUS = 1039, /\* Temperature \*/

NETDEV\_ALARM\_SMART\_REALLOCATED\_EVENT\_COUNT = 1040, /\* Number of reallocated events \*/

NETDEV\_ALARM\_SMART\_CURRENT\_PENDING\_SECTOR = 1041, /\* Number of pending sectors \*/

NETDEV\_ALARM\_SMART\_OFFLINE\_UNCORRECTABLE = 1042, /\* Number of sectors that are uncorrectable offline \*/

NETDEV\_ALARM\_SMART\_UDMA\_CRC\_ERROR\_COUNT = 1043, /\* Parity check error rate \*/

NETDEV\_ALARM\_SMART\_MULTI\_ZONE\_ERROR\_RATE = 1044, /\* Multi-zone error rate \*/

NETDEV\_ALARM\_RESOLUTION\_CHANGE = 1045, /\* Resolution change \*/

NETDEV\_ALARM\_MANUAL = 1401, /\* Manual alarm \*/

NETDEV\_ALARM\_ALARMHOST\_COMMON = 1402, /\* Alarm host event \*/

NETDEV\_ALARM\_DOORHOST\_COMMON = 1403, /\* Access control event \*/

NETDEV\_ALARM\_FACE\_NOT\_MATCH = 1411, /\* Face not match \*/

NETDEV\_ALARM\_FACE\_MATCH\_SUCCEED = 1412, /\* Face match \*/

NETDEV\_ALARM\_VIDEO\_CHL\_REACH\_UPPER\_LIMIT = 1413, /\* Number of encoding channels reached upper limit \*/

NETDEV\_ALARM\_VEHICLE\_BLACK\_LIST = 1420, /\* Vehicle recognition blocklist alarm \*/

NETDEV\_ALARM\_HUMAN\_SHAPE\_DETECTION = 1421, /\* Human body detection \*/

NETDEV\_ALARM\_HUMAN\_SHAPE\_DETECTION\_RECOVER = 1422, /\* Human body detection cleared \*/

NETDEV\_ALARM\_NOT\_WORN\_SAFETYHELMET = 1423, /\* No helmet alarm \*/

NETDEV\_ALARM\_NOT\_WORN\_WORKCLOTHES = 1424, /\* No work cloths alarm \*/

NETDEV\_ALARM\_FAST\_MOVING = 1425, /\* Quick moving alarm \*/

NETDEV\_ALARM\_NOT\_WORN\_CHEFHAT = 1426, /\* No chef hat alarm \*/

NETDEV\_ALARM\_TELEPHONING = 1427, /\* Calling alarm \*/

NETDEV\_ALARM\_SMOKING = 1428, /\* Smoking alarm \*/

NETDEV\_ALARM\_CROWD\_DENSITY\_MINOR = 1429, /\* Crowd density minor alarm \*/

NETDEV\_ALARM\_CROWD\_DENSITY\_MINOR\_CLEARED = 1430, /\* Crowd density minor alarm cleared \*/

NETDEV\_ALARM\_CROWD\_DENSITY\_MAJOR = 1431, /\* Crowd density major alarm \*/

NETDEV\_ALARM\_CROWD\_DENSITY\_MAJOR\_CLEARED = 1432, /\* Crowd density major alarm cleared \*/

NETDEV\_ALARM\_CROWD\_DENSITY\_CRITICAL = 1433, /\* Crowd density critical alarm \*/

NETDEV\_ALARM\_CROWD\_DENSITY\_CRITICAL\_CLEARED = 1434, /\* Crowd density critical alarm cleared \*/

NETDEV\_ALARM\_ACCESS\_ELEVATOR = 1435, /\* Elevator entrance alarm \*/

NETDEV\_ALARM\_ACCESS\_ELEVATOR\_CLEARED = 1436, /\* Elevator entrance alarm cleared \*/

NETDEV\_ALARM\_INVALID = 0xFFFF /\* Invalid value\*/

}NETDEV\_ALARM\_TYPE\_E;

### Enumeration of face search types

typedef enum tagNETDEVSearchType

{

NETDEV\_SEARCH\_TYPE\_FACE = 0, /\* Face search \*/

NETDEV\_SEARCH\_TYPE\_FACE\_COMPARE\_SUCCESS = 1, /\* Face comparison succeeded \*/

NETDEV\_SEARCH\_TYPE\_FACE\_COMPARE\_FAIL = 2, /\* Face comparison failed \*/

NETDEV\_SEARCH\_TYPE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_SEARCH\_TYPE\_E;

### Enumeration of organization types

typedef enum tagNETDEVOrgType

{

NETDEV\_ORG\_TYPE\_GENERAL = 0, /\* General organization \*/

NETDEV\_ORG\_TYPE\_CLOUD = 1, /\* Cloud organization \*/

NETDEV\_ORG\_TYPE\_VIRTUAL = 2, /\* Virtual organization \*/

NETDEV\_ORG\_TYPE\_FAVORITES = 3, /\* Favorites \*/

NETDEV\_ORG\_TYPE\_DOMAIN = 4, /\* Domain \*/

NETDEV\_ORG\_TYPE\_DOORGROUP = 5, /\* Door group \*/

NETDEV\_ORG\_TYPE\_DEPT = 6, /\* Department \*/

NETDEV\_ORG\_TYPE\_APARTMENT = 7, /\* Apartment \*/

NETDEV\_ORG\_TYPE\_INVALID = 0XFF /\* Invalid value \*/

}NETDEV\_ORG\_TYPE\_E;

### Enumeration of modeling statuses

typedef enum tagNETDEVModelStatus

{

NETDEV\_MODEL\_STATUS\_TYPE\_UNMODELED = 0, /\* 0: Modeling not started \*/

NETDEV\_MODEL\_STATUS\_TYPE\_SUCCEED = 1, /\* 1: Modeling completed \*/

NETDEV\_MODEL\_STATUS\_TYPE\_FAILED = 2, /\* 2: Modeling failed \*/

NETDEV\_MODEL\_STATUS\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_MODEL\_STATUS\_E;

### Enumeration of mask wearing statuses

typedef enum tagNETDEVMaskFlag

{

NETDEV\_MASK\_FLAG\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_MASK\_FLAG\_NOT\_WEAR = 1, /\* No mask \*/

NETDEV\_MASK\_FLAG\_WEAR = 2, /\* With mask \*/

NETDEV\_MASK\_FLAG\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_MASK\_FLAG\_E;

### Enumeration of logic types of search criteria

typedef enum tagNETDEVQueryCondLogic

{

NETDEV\_QUERYCOND\_LOGIC\_EQUAL = 0, /\* Equal to \*/

NETDEV\_QUERYCOND\_LOGIC\_GREATER = 1, /\* Greater than \*/

NETDEV\_QUERYCOND\_LOGIC\_LESS = 2, /\* Less than \*/

NETDEV\_QUERYCOND\_LOGIC\_NO\_LESS = 3, /\* No less than \*/

NETDEV\_QUERYCOND\_LOGIC\_NO\_GREATER = 4, /\* No greater than \*/

NETDEV\_QUERYCOND\_LOGIC\_NO\_EQUAL = 5, /\* Not equal to \*/

NETDEV\_QUERYCOND\_LOGIC\_DIM\_QUERY = 6, /\* Fuzzy search \*/

NETDEV\_QUERYCOND\_LOGIC\_CONTAIN = 7, /\* Include \*/

NETDEV\_QUERYCOND\_LOGIC\_ASC\_ORDER = 8, /\* Ascending order \*/

NETDEV\_QUERYCOND\_LOGIC\_DESC\_ORDER = 9 /\* Descending order \*/

}NETDEV\_QUERYCOND\_LOGICTYPE\_E;

### Enumeration of alarm sub types

typedefenumtagNETDEVAlarmSubType

{

NETDEV\_ALARM\_SUB\_TYPE\_NOT\_CONFIGURED =0, /\* Unconfigured alarm sub types\*/

NETDEV\_DEV\_TYPE\_BASE =100000UL, /\* Alarm base \*/

NETDEV\_DEV\_TROUBLE =100001, /\*Device failure\*/

NETDEV\_DEV\_ALARM\_EMERGENCY =100002, /\*Device emergency alarm \*/

/\*Definition of alarm types of alarm devices \*/

NETDEV\_ALARM\_EVENT\_BASE =100200UL, /\*Start of definitions of alarm types \*/

NETDEV\_ALARM\_EVENT\_FIRE\_ALARM =100201, /\*Fire alarm \*/

NETDEV\_ALARM\_EVENT\_KEYPADFIRE =100202, /\*Keypad fire alarm \*/

NETDEV\_ALARM\_EVENT\_KEYPADEMERGENCY =100203, /\* Keypad emergency alarm \*/

NETDEV\_ALARM\_EVENT\_KEYPADMEDICAL =100204, /\* Keypad help alarm \*/

NETDEV\_ALARM\_EVENT\_DURESSCODEUSED =100205, /\* Duress code operation \*/

NETDEV\_ALARM\_EVENT\_UNAUTHORIZEDENTRY =100206, /\* Unauthorized access \*/

NETDEV\_ALARM\_EVENT\_BURGLARPOINTALARM =100207, /\* Burglar zone alarm \*/

NETDEV\_ALARM\_EVENT\_SUPERVISORY =100208, /\* Detection failure \*/

NETDEV\_ALARM\_EVENT\_FIRETROUBLE =100209, /\* Fire alarm failure \*/

NETDEV\_ALARM\_EVENT\_FIREPOINTTROUBLE =100210, /\* Fire zone failure \*/

NETDEV\_ALARM\_EVENT\_BURGLARPOINTTROUBLE =100211, /\* Burglar zone failure \*/

NETDEV\_ALARM\_EVENT\_CANCELALARM =100212, /\* Cancel alarm \*/

NETDEV\_ALARM\_EVENT\_DISARM =100213, /\* Disarming succeeded \*/

NETDEV\_ALARM\_EVENT\_ARM =100214, /\* Arming succeeded \*/

NETDEV\_ALARM\_EVENT\_FORCEDBYPASS =100215, /\* Forced bypass \*/

NETDEV\_ALARM\_EVENT\_BYPASS =100216, /\* Bypass \*/

NETDEV\_ALARM\_EVENT\_POINTRESTORAL =100217, /\* Zone alarm cleared \*/

NETDEV\_ALARM\_EVENT\_FAILTOCLOSE =100218, /\* Arming failed \*/

NETDEV\_ALARM\_EVENT\_FAILTOOPEN =100219, /\* Disarming failed \*/

NETDEV\_ALARM\_EVENT\_HISTORYBUFFERDUMP =100220, /\* Event records full \*/

NETDEV\_ALARM\_EVENT\_TESTMSG =100221, /\* Test \*/

NETDEV\_ALARM\_EVENT\_ZONETROUBLE =100222, /\* Zone failure \*/

NETDEV\_ALARM\_EVENT\_ZONETROUBLERESTORE =100223, /\* Zone failure cleared \*/

NETDEV\_ALARM\_EVENT\_ZONESHORT =100224, /\* Zone short circuit \*/

NETDEV\_ALARM\_EVENT\_ZONEOPEN =100225, /\*Zone open circuit \*/

NETDEV\_ALARM\_EVENT\_ZONENORMAL =100226, /\* Zone normal \*/

NETDEV\_ALARM\_EVENT\_MXDEVCLOSE =100227, /\*MX device armed \*/

NETDEV\_ALARM\_EVENT\_MXDEVOPEN =100228, /\*MX device disarmed \*/

NETDEV\_ALARM\_EVENT\_MXDEV1ZONEALARM =100229, /\*MX device zone 1 alarm \*/

NETDEV\_ALARM\_EVENT\_MXDEV2ZONEALARM =100230, /\*MX device zone 2 alarm \*/

NETDEV\_ALARM\_EVENT\_MXDEV3ZONEALARM =100231, /\*MX device zone 3 alarm \*/

NETDEV\_ALARM\_EVENT\_MXDEV1ZONERESTAORE =100232, /\*MX device zone 1 alarm cleared \*/

NETDEV\_ALARM\_EVENT\_MXDEV2ZONERESTAORE =100233, /\*MX device zone 2 alarm cleared \*/

NETDEV\_ALARM\_EVENT\_MXDEV3ZONERESTAORE =100234, /\*MX device zone 3 alarm cleared \*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_AC =100235, /\* AC power failure \*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_BATTERY =100236, /\* Backup power failure \*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_SNDREPORT =100237, /\* Report sending failure \*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_CONTROL =100238, /\* Control failure \*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_MXBUS =100239, /\*MPX bus failure \*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_RADIOSND =100240, /\* Wireless receiving failure \*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_ASSISTPOWER =100241, /\* Auxiliary power failure \*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_OPTION =100242, /\* Option device failure \*/

NETDEV\_ALARM\_EVENT\_LINK\_BREAK =100243, /\* Link disconnected \*/

NETDEV\_ALARM\_EVENT\_LINK\_RESUME =100244, /\* Link normal \*/

NETDEV\_ALARM\_EVENT\_VOLTAGE\_ALARM =100245, /\* DC undervoltage alarm \*/

NETDEV\_ALARM\_EVENT\_EXTERNDEV\_TROUBLE =100246, /\* Expansion device failure alarm \*/

NETDEV\_ALARM\_EVENT\_EXTERNDEV\_RESUME =100247, /\* Expansion device failure alarm cleared \*/

NETDEV\_ALARM\_EVENT\_POINT\_DISARM =100248, /\* Point disarmed \*/

NETDEV\_ALARM\_EVENT\_POINT\_ARM =100249, /\* Point armed \*/

NETDEV\_ALARM\_EVENT\_FORCE\_OPEN =100250, /\* Alarm control panel opened by force \*/

NETDEV\_ALARM\_EVENT\_BYPASS\_FAILED =100251, /\* Bypass failed \*/

NETDEV\_ALARM\_EVENT\_RESET\_FAILED =100252, /\* Reset failed \*/

NETDEV\_ALARM\_EVENT\_POINT\_FIRETROUBLE =100253, /\* Point failure \*/

NETDEV\_ALARM\_EVENT\_POINT\_SEPARATE =100254, /\* Point isolation \*/

NETDEV\_ALARM\_EVENT\_POINT\_ACTION =100255, /\* Point action \*/

NETDEV\_ALARM\_EVENT\_POINT\_RESUME =100256, /\* Point recovered \*/

NETDEV\_ALARM\_EVENT\_ROB\_ACTION =100257, /\* Robbery alarm \*/

/\*Newly added alarms and events \*/

NETDEV\_ALARM\_EVENT\_MISSED\_ARM =100260, /\*Not armed \*/

NETDEV\_ALARM\_EVENT\_MISSED\_DISARM =100261, /\*Not disarmed \*/

NETDEV\_ALARM\_EVENT\_AUXILARY\_RST =100262, /\* Auxiliary alarm cleared \*/

NETDEV\_ALARM\_EVENT\_AUXILARY =100263, /\* Auxiliary alarm \*/

NETDEV\_ALARM\_EVENT\_BURGLARY\_RST =100264, /\* Burglary alarm cleared \*/

NETDEV\_ALARM\_EVENT\_BURGLARY =100265, /\* Burglary alarm \*/

NETDEV\_ALARM\_EVENT\_BYPASS\_RST =100266, /\* Bypass cleared \*/

NETDEV\_ALARM\_EVENT\_CANCEL =100267, /\* Cancel \*/

NETDEV\_ALARM\_EVENT\_DUPLICATE =100268, /\* Duplicate \*/

NETDEV\_ALARM\_EVENT\_DURESS\_RST =100269, /\* Duress alarm cleared \*/

NETDEV\_ALARM\_EVENT\_DURESS =100270, /\* Duress alarm\*/

NETDEV\_ALARM\_EVENT\_FIRE\_RST =100271, /\* Fire alarm cleared \*/

NETDEV\_ALARM\_EVENT\_FIRE\_TROUBLE =100272, /\* Fire detector failure \*/

NETDEV\_ALARM\_EVENT\_FIRE\_TRB\_RST =100273, /\* Fire detector failure cleared \*/

NETDEV\_ALARM\_EVENT\_MED\_RST =100274, /\* Medical assistance reset \*/

NETDEV\_ALARM\_EVENT\_PANIC =100275, /\* Emergency alarm \*/

NETDEV\_ALARM\_EVENT\_PANIC\_RST =100276, /\* Emergency alarm cleared \*/

NETDEV\_ALARM\_EVENT\_RESET =100277, /\* Reset \*/

NETDEV\_ALARM\_EVENT\_RELAY\_SHORTED =100278, /\* Relay short circuit \*/

NETDEV\_ALARM\_EVENT\_RELAY\_RST =100279, /\* Relay reset \*/

NETDEV\_ALARM\_EVENT\_RELAY\_TROUBLE =100280, /\* Relay failure \*/

NETDEV\_ALARM\_EVENT\_TAMPER =100281, /\* Tamper alarm \*/

NETDEV\_ALARM\_EVENT\_TAMPER\_RST =100282, /\* Tamper alarm cleared \*/

NETDEV\_ALARM\_EVENT\_TROUBLE =100283, /\* Failure \*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_RST =100284, /\* Failure cleared \*/

/\*DF8900\*/

NETDEV\_ALARM\_EVENT\_DFE\_REMOTE\_CONTROL=100290, /\* Remote control \*/

NETDEV\_ALARM\_EVENT\_DFE\_SYNC =100291, /\* Sync \*/

NETDEV\_ALARM\_EVENT\_DFE\_LIFTING =100292, /\* Lifting \*/

NETDEV\_ALARM\_EVENT\_DFE\_SWITCH\_NORMAL =100293, /\* Normal switch change \*/

NETDEV\_ALARM\_EVENT\_DFE\_SWITCH\_ACCIDENT=100294, /\* Accident switch change \*/

NETDEV\_ALARM\_EVENT\_DFE\_KINFE\_GATE =100295, /\* Normal knife switch change \*/

NETDEV\_ALARM\_EVENT\_ESCAPE =100296, /\* ChenYingShaoWei event

NETDEV\_ALARM\_EVENT\_ESCAPE\_END =100297, /\* ChenYingShaoWei event

NETDEV\_ALARM\_EVENT\_VIOLENCE =100298, /\* ChenYingShaoWei event

NETDEV\_ALARM\_EVENT\_VIOLENCE\_END =100299, /\* ChenYingShaoWei event

NETDEV\_ALARM\_EVENT\_ATTACKED =100300, /\* ChenYingShaoWei event

NETDEV\_ALARM\_EVENT\_ATTACKED\_END =100301, /\* ChenYingShaoWei event

NETDEV\_ALARM\_EVENT\_NATURAL\_DISASTER =100302, /\* ChenYingShaoWei event

NETDEV\_ALARM\_EVENT\_NATURAL\_DISASTER\_END =100303, /\* ChenYingShaoWei event

NETDEV\_ALARM\_EVENT\_ONE\_KEY\_ALARM =100304, /\* ChenYingShaoWei event

NETDEV\_ALARM\_EVENT\_ONE\_KEY\_ALARM\_END =100305, /\* ChenYingShaoWei event

NETDEV\_ALARM\_EVENT\_CERC\_NORMAL =100306, /\* ZhongDianRuiDa event

NETDEV\_ALARM\_EVENT\_CERC\_TOUCH =100307, /\* ZhongDianRuiDa event

NETDEV\_ALARM\_EVENT\_CERC\_BREAK =100308, /\* ZhongDianRuiDa event

NETDEV\_ALARM\_EVENT\_CERC\_SHORT =100309, /\* ZhongDianRuiDa event

NETDEV\_ALARM\_EVENT\_CERC\_COMMU\_ERR =100310, /\* ZhongDianRuiDa event

,

NETDEV\_ALARM\_EVENT\_HYPERSEE\_DDR\_ALARM =100311, /\* AiBoXin event

NETDEV\_ALARM\_EVENT\_HYPERSEE\_DDR\_ALARM\_RST =100312, /\*AiBoXin event

NETDEV\_ALARM\_EVENT\_HYPERSEE\_IR\_ALARM =100313, /\*AiBoXin event

NETDEV\_ALARM\_EVENT\_HYPERSEE\_IR\_ALARM\_RST =100314, /\*AiBoXin event

NETDEV\_ALARM\_EVENT\_UBI\_ALARM =100315, /\*ZhongKeRongTong event

NETDEV\_ALARM\_EVENT\_HXTECH\_CABLE\_ALARM =100316, /\*HuaXun perimeter event

NETDEV\_ALARM\_EVENT\_DEVICE\_UNDER\_VOLTAGE\_RST =100317, /\* Device undervoltage cleared\*/

NETDEV\_ALARM\_EVENT\_ZONE\_NOT\_READY =100318, /\* Zone not ready \*/

NETDEV\_ALARM\_EVENT\_ZONE\_NOT\_READY\_RST =100319, /\* Zone not ready cleared \*/

NETDEV\_ALARM\_EVENT\_TELEPHONE\_CABLE\_FAULT =100320, /\*Telephone cable fault \*/

NETDEV\_ALARM\_EVENT\_TELEPHONE\_CABLE\_FAULT\_RST=100321, /\*Telephone cable fault cleared\*/

NETDEV\_ALARM\_EVENT\_AC\_POWER\_FAULT =100322, /\* AC power fault \*/

NETDEV\_ALARM\_EVENT\_ZONE\_TOUCH =100323, /\* Zone trigger alarm \*/

NETDEV\_ALARM\_EVENT\_ZONE\_TOUCH\_RESTORE =100324, /\* Zone trigger alarm cleared \*/

NETDEV\_ALARM\_EVENT\_CAPTIVE =100325, /\* Force open \*/

NETDEV\_ALARM\_EVENT\_CAPTIVE\_RST =100326, /\* Force open cleared \*/

NETDEV\_ALARM\_EVENT\_UNDERVOLTAGE =100327, /\* Undervoltage \*/

NETDEV\_ALARM\_EVENT\_UNDERVOLTAGE\_RST =100328, /\* Undervoltage cleared \*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_BATTERY\_RST =100329, /\* Backup battery fault cleared \*/

NETDEV\_ALARM\_EVENT\_DEV\_POWER\_REBOOT =100330, /\* Device power-on restart \*/

NETDEV\_ALARM\_EVENT\_TEST\_FAILURE =100331, /\* Device test failure \*/

NETDEV\_ALARM\_EVENT\_OPERATOR\_TOUCH\_LONG =100332, /\* Long press operation \*/

NETDEV\_ALARM\_EVENT\_REMOTE\_ARM =100333, /\* Remote arm \*/

NETDEV\_ALARM\_EVENT\_REMOTE\_DISARM =100334, /\* Remote disarm \*/

NETDEV\_ALARM\_EVENT\_REMOTE\_STAY\_ARM =100335, /\* Remote arm stay \*/

NETDEV\_ALARM\_EVENT\_PWD\_ARM =100336, /\* Password arm device \*/

NETDEV\_ALARM\_EVENT\_PWD\_DISARM =100337, /\* Password disarm device \*/

NETDEV\_ALARM\_EVENT\_PWD\_STAY\_DEV =100338, /\* Password arm device stay \*/

NETDEV\_ALARM\_EVENT\_PWD\_ARM\_PART =100339, /\* Password arm partition \*/

NETDEV\_ALARM\_EVENT\_PWD\_DISARM\_PART =100340, /\*Password disarm partition \*/

NETDEV\_ALARM\_EVENT\_PWD\_STAY\_PART =100341, /\* Password arm partition stay \*/

NETDEV\_ALARM\_EVENT\_PWD\_ARM\_ZONE =100342, /\* Password arm zone \*/

NETDEV\_ALARM\_EVENT\_PWD\_DISARM\_ZONE =100343, /\* Password disarm zone \*/

NETDEV\_ALARM\_EVENT\_USER\_ARM =100344, /\* User arm device \*/

NETDEV\_ALARM\_EVENT\_USER\_DISARM =100345, /\* User disarm device \*/

NETDEV\_ALARM\_EVENT\_USER\_STAY\_DEV =100346, /\* User arm device stay \*/

NETDEV\_ALARM\_EVENT\_USER\_CLEAN\_ALARM =100347, /\* User clear alarm display \*/

NETDEV\_ALARM\_EVENT\_PWD\_OPEN\_DOOR =100348, /\* Open door by password \*/

NETDEV\_ALARM\_EVENT\_KEY\_OPEN\_DOOR =100349, /\* Open door by key \*/

NETDEV\_ALARM\_EVENT\_REMOTE\_OPEN\_DOOR =100350, /\* Open door remotely \*/

NETDEV\_ALARM\_EVENT\_SWIPE\_ARM =100351, /\* Arm by card swipe \*/

NETDEV\_ALARM\_EVENT\_SWIPE\_DISARM =100352, /\* Disarm by card swipe \*/

NETDEV\_ALARM\_EVENT\_SWIPE\_LOCK =100353, /\* Lock by card swipe \*/

NETDEV\_ALARM\_EVENT\_SWIPE\_UNLOCK =100354, /\* Unlock by card swipe \*/

NETDEV\_ALARM\_EVENT\_SWIPE\_RECORD =100355, /\* Card swipe record \*/

NETDEV\_ALARM\_EVENT\_SWIPE\_ATTENDANCE\_IN=100356, /\* Attendance in by card swipe \*/

NETDEV\_ALARM\_EVENT\_SWIPE\_ATTENDANCE\_OUT=100357, /\* Attendance out by card swipe \*/

NETDEV\_ALARM\_EVENT\_SWIPE\_TURNON\_LIGHT =100358, /\* Turn on light by card swipe \*/

NETDEV\_ALARM\_EVENT\_SWIPE\_OFF\_LIGHT =100359, /\* Turn off light by card swipe \*/

NETDEV\_ALARM\_EVENT\_SWIPE\_CARD\_NUM =100360, /\* Report card physical number by card swipe \*/

NETDEV\_ALARM\_EVENT\_MANUAL\_OPEN =100361, /\* Open door or turn on light manually \*/

NETDEV\_ALARM\_EVENT\_ZONE\_OPEN\_DOOR =100362, /\* Open door by zone trigger \*/

NETDEV\_ALARM\_EVENT\_ZONE\_DOOR\_LONGTIME=100363, /\* Zone long time no trigger arm\*/

NETDEV\_ALARM\_EVENT\_ZONE\_ARM =100364, /\* Zone trigger arm\*/

NETDEV\_ALARM\_EVENT\_ZONE\_DISARM =100365, /\* Zone trigger disarm \*/

NETDEV\_ALARM\_EVENT\_TIMER\_ARM =100366, /\* Scheduled arm \*/

NETDEV\_ALARM\_EVENT\_TIMER\_DISARM =100367, /\* Scheduled disarm \*/

NETDEV\_ALARM\_EVENT\_CENTRAL\_COMM\_NORMAL =100368, /\* Central communication normal \*/

NETDEV\_ALARM\_EVENT\_CENTRAL\_HANDSHAKE\_FAILED =100369, /\* Central handshake failure \*/

NETDEV\_ALARM\_EVENT\_CENTRAL\_RESPONSE\_FAILED =100370, /\* Central response failure \*/

NETDEV\_ALARM\_EVENT\_USER\_CONFIRMATION =100371, /\* User confirmed alarm \*/

NETDEV\_ALARM\_EVENT\_USER\_TIMEOUT =100372, /\* Alarm confirmation timed out \*/

NETDEV\_ALARM\_EVENT\_TELEP\_RINGING =100373, /\* Telephone module receive ringing signal \*/

NETDEV\_ALARM\_EVENT\_TELEP\_ARM =100374, /\* Telephone module receive 1-digit remote disarm/arm password \*/

NETDEV\_ALARM\_EVENT\_TELEP\_STATUS\_OFFLINE =100375, /\* Telephone offline \*/

NETDEV\_ALARM\_EVENT\_TELEP\_STATUS\_ONLINE =100376, /\* Telephone online \*/

NETDEV\_ALARM\_EVENT\_MSG\_SEND\_SUCC =100377, /\*MSG sending succeeded \*/

NETDEV\_ALARM\_EVENT\_MSG\_SEND\_FAILED =100378, /\*MSG sending failed \*/

NETDEV\_ALARM\_EVENT\_SWITCH\_CLOSE =100379, /\* Operation switch close \*/

NETDEV\_ALARM\_EVENT\_SWITCH\_OFF =100380, /\* Operation switch open \*/

NETDEV\_ALARM\_EVENT\_TEMPER\_CHANGE =100381, /\* Temperature change \*/

NETDEV\_ALARM\_EVENT\_HUMIDITY\_CHANGE =100382, /\*Humidity change \*/

NETDEV\_ALARM\_EVENT\_DEVID\_WRONG =100383, /\* Device ID error \*/

NETDEV\_ALARM\_EVENT\_ELEC\_HIGH\_ARM =100384, /\*Single electric fence high voltage arm\*/

NETDEV\_ALARM\_EVENT\_ELEC\_LOW\_ARM =100385, /\*Single electric fence low voltage arm \*/

NETDEV\_ALARM\_EVENT\_ELEC\_ALARM\_RST =100386, /\* Single electric fence reset \*/

NETDEV\_ALARM\_EVENT\_ELEC\_DISARM =100387, /\* Single electric fence disarm \*/

NETDEV\_ALARM\_EVENT\_ELEC\_ARM =100388, /\* Single electric fence voltage value arm \*/

NETDEV\_ALARM\_EVENT\_KEY\_ALARM\_OVER =100389, /\* Keypad all alarms handled \*/

NETDEV\_ALARM\_EVENT\_KEY\_PROL\_TYPE =100390, /\* Keypad report protocol type (mainly for electric fence) \*/

NETDEV\_ALARM\_EVENT\_MOD\_DEV\_ADDR =100391, /\* Keypad change device address (mainly for electric fence) \*/

NETDEV\_ALARM\_EVENT\_FORWARD\_CMD =100392, /\* Communication device forward central command \*/

NETDEV\_ALARM\_EVENT\_BASE\_VALUE =110000UL, /\* Start of definitions of alarm types \*/

NETDEV\_ALARM\_EVENT\_IMMEDIATELY\_DEFEND\_ALARM =110001, /\* Instant zone alarm \*/

NETDEV\_ALARM\_EVENT\_IMMEDIATELY\_DEFEND\_ALARM\_RST=110002, /\* Instant zone alarm cleared \*/

NETDEV\_ALARM\_EVENT\_ALL\_DAY\_NO\_VOICE\_ALARM =110003, /\*24-hour silent zone alarm \*/

NETDEV\_ALARM\_EVENT\_ALL\_DAY\_NO\_VOICE\_ALARM\_RST=110004, /\*24-hour silent zone alarm cleared \*/

NETDEV\_ALARM\_EVENT\_ALL\_DAY\_VOICE\_ALARM =110005, /\*24-hour voiced zone alarm \*/

NETDEV\_ALARM\_EVENT\_ALL\_DAY\_VOICE\_ALARM\_RST =110006, /\*24-hour voiced zone alarm cleared \*/

NETDEV\_ALARM\_EVENT\_BOARD\_DEFEND\_ALARM =110007, /\* Perimeter zone alarm \*/

NETDEV\_ALARM\_EVENT\_BOARD\_DEFEND\_ALARM\_RST =110008, /\*Perimeter zone alarm cleared \*/

NETDEV\_ALARM\_EVENT\_INNER\_DELAY\_DEFEND\_ALARM =110009, /\* Internal delay zone alarm \*/

NETDEV\_ALARM\_EVENT\_INNER\_DELAY\_DEFEND\_ALARM\_RST=110010, /\* Internal delay zone alarm cleared \*/

NETDEV\_ALARM\_EVENT\_DELAY\_DEFEND\_ALARM =110011, /\* Delay zone alarm \*/

NETDEV\_ALARM\_EVENT\_DELAY\_DEFEND\_ALARM\_RST =110012, /\*Delay zone alarm cleared \*/

NETDEV\_ALARM\_EVENT\_EXPAND\_MODEL\_FAULT\_ALARM=110013, /\* Expansion module failure \*/

NETDEV\_ALARM\_EVENT\_EXPAND\_MODEL\_FAULT\_ALARM\_RST=110014, /\* Expansion module failure cleared \*/

NETDEV\_ALARM\_EVENT\_OUTER\_DISARMED =110015, /\* Disarm away \*/

NETDEV\_ALARM\_EVENT\_OUTER\_ARMED =110016, /\* Armed away \*/

NETDEV\_ALARM\_EVENT\_KEY\_DISARMED =110017, /\* Keyed zone disarmed \*/

NETDEV\_ALARM\_EVENT\_KEY\_ARMED =110018, /\* Keyed zone armed \*/

NETDEV\_ALARM\_EVENT\_WIRELESS\_NET\_WORK\_EXCEPTION =110019, /\* Wireless network abnormal \*/

NETDEV\_ALARM\_EVENT\_WIRELESS\_NET\_WORK\_EXCEPTION\_RST=110020, /\* Wireless network abnormal cleared \*/

NETDEV\_ALARM\_EVENT\_WIRED\_NET\_WORK\_BREAK =110021, /\* Wired network failure \*/

NETDEV\_ALARM\_EVENT\_WIRED\_NET\_WORK\_BREAK\_RST =110022, /\* Wired network failure cleared \*/

NETDEV\_ALARM\_EVENT\_SOFT\_DEFEND\_URGENCY\_ALARM =110023, /\* Soft zone emergency alarm \*/

NETDEV\_ALARM\_EVENT\_ARMED\_STAY =110024, /\*Armed stay \*/

NETDEV\_ALARM\_EVENT\_IMMEDIATELY\_ARMED =110025, /\* Instant arming \*/

NETDEV\_ALARM\_EVENT\_SMOKE\_ALARM =110026, /\* Smoke alarm \*/

NETDEV\_ALARM\_EVENT\_SMOKE\_ALARM\_END =110027, /\* Smoke alarm ended \*/

NETDEV\_ALARM\_EVENT\_WATER\_ALARM =110028, /\* Water alarm \*/

NETDEV\_ALARM\_EVENT\_WATER\_ALARM\_END =110029, /\* Water alarm ended \*/

NETDEV\_ALARM\_EVENT\_IO\_ON =110030, /\* I/O on\*/

NETDEV\_ALARM\_EVENT\_IO\_OFF =110031, /\*IO off\*/

NETDEV\_ALARM\_EVENT\_TOUNCH\_ALARM =110032, /\* Touch alarm \*/

NETDEV\_ALARM\_EVENT\_INVADE\_ALARM =110033, /\* Intrusion alarm \*/

NETDEV\_ALARM\_EVENT\_BROKEN\_ALARM =110034, /\* Broken alarm \*/

NETDEV\_ALARM\_EVENT\_HELP =110036, /\* Emergent help \*/

NETDEV\_ALARM\_EVENT\_DESTROY =110037, /\* Vandalism alarm \*/

NETDEV\_ALARM\_EVENT\_ACS =110038, /\* Access control alarm \*/

NETDEV\_ALARM\_EVENT\_OTHERS =110039, /\*Other\*/

NETDEV\_ALARM\_EVENT\_ZONE\_ALARM =110040, /\*Zone alarm \*/

NETDEV\_ALARM\_EVENT\_ZONE\_ALARM\_RST =110041, /\* Zone alarm cleared \*/

NETDEV\_ALARM\_EVENT\_GATO\_ALARM\_FENCE =110042, /\* Fence alarm \*/

NETDEV\_ALARM\_EVENT\_GATO\_ALARM\_HOST =110043, /\*Control panel alarm \*/

NETDEV\_ALARM\_EVENT\_GATO\_ALARM\_ETCZONE =110044, /\* Expansion zone alarm \*/

NETDEV\_ALARM\_EVENT\_GATO\_ALARM\_OFFLINE =110045, /\* Communication disconnected \*/

NETDEV\_ALARM\_EVENT\_GATO\_ALARM\_SWITCH1 =110046, /\* Zone alarm input 1\*/

NETDEV\_ALARM\_EVENT\_GATO\_ALARM\_SWITCH2 =110047, /\* Zone alarm input 2\*/

NETDEV\_ALARM\_EVENT\_STEAL =110048, /\*Theft alarm \*/

NETDEV\_ALARM\_EVENT\_GAS =110049, /\* Gas alarm \*/

NETDEV\_ALARM\_EVENT\_SAVE =110050, /\*Rescue alarm \*/

NETDEV\_ALARM\_EVENT\_INVADE\_TROUBLE =110051, /\* Intrusion & failure alarm \*/

NETDEV\_ALARM\_EVENT\_SYS\_TROUBLE =110052, /\* System failure \*/

NETDEV\_ALARM\_EVENT\_OPTICAL\_FIBER\_TROUBLE =110053, /\* Optical fiber failure \*/

NETDEV\_ALARM\_EVENT\_MOTION\_DETECTION =110062, /\* Motion detection alarm \*/

NETDEV\_ALARM\_EVENT\_CALL\_REMOVE =110063, /\* Call cancelled alarm \*/

NETDEV\_ALARM\_EVENT\_NOISE =110064, /\* Noise alarm \*/

NETDEV\_ALARM\_EVENT\_DELAYED =110065, /\* Delayed alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_INPUT\_ONE =110066, /\* Alarm input 1 alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_INPUT\_TWO =110067, /\* Alarm input 2 alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_CALL\_EMERGENCY =110068, /\*Emergency call \*/

NETDEV\_ALARM\_EVENT\_ALARM\_CALL\_CONSULTANT=110069, /\* consultation call \*/

NETDEV\_ALARM\_EVENT\_DOOR\_MAGNET\_OPEN =110070, /\* Door magnet opened \*/

NETDEV\_ALARM\_EVENT\_DOOR\_MAGNET\_CLOSE =110071, /\* Door magnet closed \*/

NETDEV\_ALARM\_EVENT\_DOOR\_LOCK\_OPEN =110072, /\* Door lock opened \*/

NETDEV\_ALARM\_EVENT\_DOOR\_LOCK\_CLOSE =110073, /\* Door lock closed \*/

NETDEV\_ALARM\_EVENT\_ALARM\_PORT\_ONE =110074, /\* Port 1 alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_PORT\_TWO =110075, /\* Port 2 alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_PORTEX =110076, /\* Expansion port alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_TUMULT =110077, /\* Tumult alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_110 =110078, /\*110 alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_119 =110079, /\*119 alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_120 =110080, /\*120 alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_FINDERCONNECT\_FAILED =110081, /\* Detector communication failure \*/

NETDEV\_ALARM\_EVENT\_ALARM\_OUTAGE =110082, /\* Power outage alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_DISASSEMBLE =110083, /\* Disassemble alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_CIRCUIT =110084, /\* Circuit alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_TOUCHNET =110085, /\* Touch net alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_PREVENTCUT =110086, /\* Prevent cut alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_COCKING =110087, /\* Cocking alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_SLACK =110088, /\* Slack alarm \*/

NETDEV\_ALARM\_EVENT\_ALARM\_ZONE\_FORBIDDEN =110089, /\* Zone forbidden \*/

NETDEV\_ALARM\_EVENT\_HIGH\_TEMP =110090, /\*Temperature above upper limit alarm \*/

NETDEV\_ALARM\_EVENT\_LOW\_TEMP =110091, /\*Temperature below lower limit alarm \*/

NETDEV\_ALARM\_EVENT\_HIGH\_HUMI =110092, /\* Humidity above upper limit alarm \*/

NETDEV\_ALARM\_EVENT\_LOW\_HUMI =110093, /\* Humidity below lower limit alarm \*/

NETDEV\_ALARM\_MONITOR\_DEVICE\_ONLINE =110094, /\* Monitoring device online \*/

NETDEV\_ALARM\_MONITOR\_DEVICE\_OFFLINE =110095, /\* Monitoring device offline \*/

NETDEV\_ALARM\_EVENT\_ALARM\_CALL\_OUT\_RING =110110, /\*Call out ringing \*/

NETDEV\_ALARM\_EVENT\_ALARM\_CALL\_IN =110111, /\* Call in \*/

NETDEV\_ALARM\_EVENT\_ALARM\_CALL\_END =110112, /\*Call ended \*/

NETDEV\_ALARM\_EVENT\_ALARM\_BROADCAST\_TO\_SLAVE\_BEGIN =110113,/\* Broadcast to host begin \*/

NETDEV\_ALARM\_EVENT\_ALARM\_BROADCAST\_TO\_MAST\_BEGIN =110114,/\* Broadcast to extension begin \*/

NETDEV\_ALARM\_EVENT\_ALARM\_BROADCAST\_END =110115, /\* Broadcast ended \*/

NETDEV\_ALARM\_EVENT\_ALARM\_CALL\_INTERPHONE =110116, /\* Two-way audio connected \*/

/\* Definition of access control events \*/

NETDEV\_DOOR\_EVENT\_BASE =100400UL, /\* Start of definitions of access control events \*/

NETDEV\_DOOR\_EVENT\_DISCONNECT =100401, /\* Communication Interrupted

NETDEV\_DOOR\_EVENT\_INFRARED\_ALARM =100402, /\* IR alarm \*/

NETDEV\_DOOR\_EVENT\_ILLICIT\_OPEN\_DOOR =100403, /\* Door opened illegally \*/

NETDEV\_DOOR\_EVENT\_COERCE\_ALARM =100404, /\* Coercing alarm \*/

NETDEV\_DOOR\_EVENT\_OPEN\_TIMEOUT =100405, /\* Door not opened before timeout \*/

NETDEV\_DOOR\_EVENT\_CLOSE\_TIMEOUT =100406, /\* Door not closed before timeout \*/

NETDEV\_DOOR\_EVENT\_ANOMALOUS\_CARD =100407, /\* Abnormal card entry \*/

NETDEV\_DOOR\_EVENT\_CARD\_OPEN\_DOOR =100408, /\* Door opened by card \*/

NETDEV\_DOOR\_EVENT\_REMOTE\_OPEN\_DOOR =100409, /\* Door opened remotely \*/

NETDEV\_DOOR\_EVENT\_MANUAL\_OPEN\_DOOR =100410, /\* Door opened manually \*/

NETDEV\_DOOR\_EVENT\_EMERGENCY\_START =100411, /\* Emergency started \*/

NETDEV\_DOOR\_EVENT\_EMERGENCY\_STOP =100412, /\* Emergency ended \*/

NETDEV\_DOOR\_EVENT\_OPEN\_SUCCESS =100413, /\* Door opened successfully \*/

NETDEV\_DOOR\_EVENT\_OPEN\_FAILED =100414, /\* Door failed to open \*/

NETDEV\_DOOR\_EVENT\_CLOSE =100415, /\* Door closing \*/

NETDEV\_DOOR\_EVENT\_CLOSE\_FAILED =100416, /\* Door failed to close \*/

NETDEV\_DOOR\_EVENT\_CARD\_NO\_AUTH =100417, /\* Unauthorized card \*/

NETDEV\_DOOR\_EVENT\_NO\_DATE\_LIMIT =100418, /\* No expiration date \*/

NETDEV\_DOOR\_EVENT\_CARD\_ILLEGAL =100419, /\* Illegal card \*/

NETDEV\_DOOR\_EVENT\_FIREPORTECT\_ALRAM =100420, /\* Fire alarm \*/

NETDEV\_DOOR\_EVENT\_CANCEL\_FIREPORTECT =100421, /\* Cancel fire alarm \*/

NETDEV\_DOOR\_EVENT\_PRYING\_RESIST\_ALRAM =100422, /\* Prying alarm \*/

NETDEV\_DOOR\_EVENT\_SUPERPASSWD\_OPEN\_DOOR =100423, /\* Door opened by super password \*/

NETDEV\_DOOR\_EVENT\_COERCECODE\_IN =100424, /\* Duress code entry \*/

NETDEV\_DOOR\_EVENT\_DOOR\_STATUS\_OPEN =100425, /\* Door opened \*/

NETDEV\_DOOR\_EVENT\_DOOR\_STATUS\_CLOSE =100426, /\* Door closed \*/

NETDEV\_DOOR\_EVENT\_DOOR\_FORCED\_OPEN =100427, /\* Door opened by force \*/

NETDEV\_DOOR\_EVENT\_DOOR\_STATUS\_OPEN =100428, /\* Door keep open \*/

NETDEV\_DOOR\_EVENT\_LOCK\_FAILURE =100429, /\* Lock failure \*/

NETDEV\_DOOR\_EVENT\_LOST\_STOLEN\_CARD =100430, /\* Lost or stolen card \*/

NETDEV\_DOOR\_EVENT\_PASSWD\_WRONG =100431, /\* Incorrect password \*/

NETDEV\_DOOR\_EVENT\_ALWAYS\_CLOSED =100432, /\* Always closed \*/

NETDEV\_DOOR\_EVENT\_CARD\_PASSWD\_ENTER =100433, /\* Card & password entry \*/

NETDEV\_DOOR\_EVENT\_CARD\_PASSWD\_WRONG =100434, /\* Wrong card & password \*/

NETDEV\_DOOR\_EVENT\_MORE\_CARD\_OPEN =100435, /\* Door opened by multiple cards \*/

NETDEV\_DOOR\_EVENT\_FIRST\_CARD\_OPEN =100436, /\* Door opened by first card \*/

NETDEV\_DOOR\_EVENT\_CARD\_OUT\_DATE =100437, /\* Card expired \*/

NETDEV\_DOOR\_EVENT\_CARD\_WATCH =100438, /\* Patrol card \*/

NETDEV\_DOOR\_EVENT\_REMOTE\_CLOSE\_DOOR =100439, /\* Door closed remotely \*/

NETDEV\_DOOR\_EVENT\_REMOTE\_ALWAYS\_OPEN =100440, /\* Remote always open \*/

NETDEV\_DOOR\_EVENT\_REMOTE\_ALWAYS\_CLOSE =100441, /\* Remote always closed \*/

NETDEV\_DOOR\_EVENT\_LEGAL\_CARD\_PASS =100442, /\* Legal card verified\*/

NETDEV\_DOOR\_EVENT\_CARD\_AND\_PSW\_PASS =100443, /\* Card & password verified \*/

NETDEV\_DOOR\_EVENT\_CARD\_AND\_PSW\_FAIL =100444, /\* Card & password verification failed）\*/

NETDEV\_DOOR\_EVENT\_CARD\_AND\_PSW\_TIMEOUT =100445, /\* Card & password verification timeout \*/

NETDEV\_DOOR\_EVENT\_CARD\_AND\_PSW\_OVER\_TIME=100446, /\* Card & password attempts reached limit\*/

NETDEV\_DOOR\_EVENT\_INVALID\_CARD =100447, /\* Card number not exist \*/

NETDEV\_DOOR\_EVENT\_DOOR\_BUTTON\_PRESS =100448, /\*Door button pressed \*/

NETDEV\_DOOR\_EVENT\_DOOR\_BUTTON\_RELEASE =100449, /\*Door button released \*/

NETDEV\_DOOR\_EVENT\_CARD\_READER\_DESMANTLE\_ALARM =100450, /\*Card reader tamper alarm \*/

NETDEV\_DOOR\_EVENT\_CARD\_READER\_DESMANTLE\_RESUME =100451, /\*Card reader tamper alarm cleared \*/

NETDEV\_DOOR\_EVENT\_DOOR\_MAGNET\_OPEN =100452, /\* Door magnet opened normally \*/

NETDEV\_DOOR\_EVENT\_DOOR\_MAGNET\_CLOSE =100453, /\* Door magnet closed normally \*/

NETDEV\_DOOR\_EVENT\_DOOR\_MAGNET\_OPEN\_ABNORMAL=100454, /\* Door magnet opened abnormally \*/

NETDEV\_DOOR\_EVENT\_DOOR\_MAGNET\_OPEN\_TIMEOUT =100455, /\* Door magnet not opened before timeout

NETDEV\_DOOR\_EVENT\_ANTI\_SNEAK\_FAIL =100456, /\* Anti-passback verification failed \*/

NETDEV\_DOOR\_EVENT\_INTERLOCK\_DOOR\_NOT\_CLOSE =100457, /\* Interlock door not closed \*/

NETDEV\_DOOR\_EVENT\_MULTI\_VERIFY\_SUCCESS =100458, /\* Multi-factor authentication succeeded \*/

NETDEV\_DOOR\_EVENT\_LEADER\_CARD\_OPEN\_BEGIN =100459, /\* Door opened by main card started \*/

NETDEV\_DOOR\_EVENT\_LEADER\_CARD\_OPEN\_END =100460, /\*Door opened by main card ended \*/

NETDEV\_DOOR\_EVENT\_ALWAYS\_OPEN\_BEGIN =100461, /\*Always open started \*/

NETDEV\_DOOR\_EVENT\_ALWAYS\_OPEN\_END =100462, /\*Always open ended \*/

NETDEV\_DOOR\_EVENT\_ALWAYS\_CLOSE\_BEGIN =100463, /\* Always close started \*/

NETDEV\_DOOR\_EVENT\_ALWAYS\_CLOSE\_END =100464, /\*Always close ended \*/

NETDEV\_DOOR\_EVENT\_KEYPRESS\_OPEN\_DOOR =100465, /\* Door opened by button \*/

NETDEV\_DOOR\_EVENT\_COMMU\_INTERRUPT =100481, /\*Communication interrupted \*/

NETDEV\_DOOR\_EVENT\_COMMU\_RESTORE =100482, /\*Communication restored\*/

NETDEV\_DOOR\_EVENT\_LEGAL\_CREDIT\_CARD =100483, /\* Legal card \*/

NETDEV\_DOOR\_EVENT\_LEGAL\_USER\_ID =100484, /\*Legal user ID \*/

NETDEV\_DOOR\_EVENT\_CARD\_NO\_REG =100485, /\* Card not registered \*/

NETDEV\_DOOR\_EVENT\_CARD\_UNAUTH =100486, /\* Card unauthorized \*/

NETDEV\_DOOR\_EVENT\_CARD\_DISABLED =100487, /\* Card disabled \*/

NETDEV\_DOOR\_EVENT\_CARD\_EXPIRED =100488, /\* Card expired \*/

NETDEV\_DOOR\_EVENT\_INVALID\_TIME =100489, /\* Invalid period \*/

NETDEV\_DOOR\_EVENT\_PSW\_ERROR =100490, /\* Incorrect password \*/

NETDEV\_DOOR\_EVENT\_PROHIBITED =100491, /\* Entry prohibited \*/

NETDEV\_DOOR\_EVENT\_REQ\_CENTER\_OPEN\_DOOR =100492, /\* Request center to open door \*/

NETDEV\_DOOR\_EVENT\_LEGAL\_CARD\_OPEN\_DOOR =100493, /\* Door opened by legal card \*/

NETDEV\_DOOR\_EVENT\_SUPER\_CARD\_OPEN\_DOOR =100494, /\*Door opened by super card \*/

NETDEV\_DOOR\_EVENT\_CENTER\_OPEN\_DOOR =100496, /\* Door opened by center \*/

NETDEV\_DOOR\_EVENT\_LINK\_OPEN\_DOOR =100497, /\* Door opened by linked action \*/

NETDEV\_DOOR\_EVENT\_FORCE\_OPEN\_DOOR =100498, /\* Door opened by force \*/

NETDEV\_DOOR\_EVENT\_DOOR\_IS\_OPEN =100499, /\* Door opened \*/

NETDEV\_DOOR\_EVENT\_DOOR\_IS\_CLOSE =100500, /\* Door closed \*/

NETDEV\_DOOR\_EVENT\_STRESS\_ALARM =100503, /\* Duress alarm \*/

NETDEV\_DOOR\_EVENT\_TRAILING\_ALARM =100505, /\* Trailing alarm \*/

NETDEV\_DOOR\_EVENT\_DOOR\_LOCK =100506, /\* Door locked \*/

NETDEV\_DOOR\_EVENT\_LIFT\_DOOR\_LOCK =100507, /\* Door unlocked \*/

NETDEV\_DOOR\_EVENT\_DOOR\_NOT\_CLOSE =100508, /\* Door not closed \*/

NETDEV\_DOOR\_EVENT\_NEED\_PSW =100509, /\* Password required \*/

NETDEV\_DOOR\_EVENT\_NEED\_FINGERPRINT =100510, /\* Fingerprint required \*/

NETDEV\_DOOR\_EVENT\_REQ\_INTERCOM =100511, /\* Intercom requested \*/

NETDEV\_DOOR\_EVENT\_EMERGENCY =100512, /\* Emergency \*/

NETDEV\_DOOR\_EVENT\_PATROL\_LEGAL\_CREDIT\_CARD =100513, /\* Legal patrol card entry \*/

NETDEV\_DOOR\_EVENT\_PATROL\_LEGAL\_FINGERPRINT =100514, /\* Legal patrol fingerprint entry \*/

NETDEV\_DOOR\_EVENT\_PATROL\_CARD\_NOT\_REG =100515, /\* Patrol card not registered \*/

NETDEV\_DOOR\_EVENT\_NOT\_PATROL\_CARD =100516, /\* Not patrol card \*/

NETDEV\_DOOR\_EVENT\_NORMAL\_PATROL =100517, /\* Normal patrol \*/

NETDEV\_DOOR\_EVENT\_ADVANCE\_PATROL =100518, /\* Patrol ahead of schedule \*/

NETDEV\_DOOR\_EVENT\_TIMEOUT\_PATROL =100519, /\* Patrol timeout \*/

NETDEV\_DOOR\_EVENT\_INVALID\_PATROL =100520, /\* Invalid patrol \*/

NETDEV\_DOOR\_EVENT\_NOT\_PATROL =100521, /\*No patrol \*/

NETDEV\_DOOR\_EVENT\_TH\_ALARM =100522, /\* Temperature & humidity alarm \*/

NETDEV\_DOOR\_EVENT\_TH\_RESTORE =100523, /\*Temperature & humidity alarm cleared \*/

NETDEV\_DOOR\_EVENT\_TH\_RECORD =100524, /\*Temperature & humidity record \*/

NETDEV\_DOOR\_EVENT\_SMOKE\_ALARM =100525, /\* Smoke alarm \*/

NETDEV\_DOOR\_EVENT\_SMOKE\_RESTORE =100526, /\* Smoke alarm cleared \*/

NETDEV\_DOOR\_EVENT\_SMOKE\_RECORD =100527, /\* Smoke record \*/

NETDEV\_DOOR\_EVENT\_PC\_CONTROL =100528, /\* Card entry denied (PC control) \*/

NETDEV\_DOOR\_EVENT\_LIMITED\_CARD\_NUM =100529, /\* Card entry denied (limited number) \*/

NETDEV\_DOOR\_EVENT\_REASON\_UNKNOWN =100530, /\* Card entry denied (unknown cause) \*/

NETDEV\_DOOR\_EVENT\_CONTROLLER\_ON =100531, /\* Controller powered on \*/

NETDEV\_DOOR\_EVENT\_CONTROLLER\_RESERT =100532, /\* Controller reset \*/

NETDEV\_DOOR\_EVENT\_DOOR\_FORCE\_CLOSE =100533, /\* Door closed by force \*/

NETDEV\_DOOR\_EVENT\_DOOR\_OFFLINE =100534, /\* Door offline \*/

NETDEV\_DOOR\_EVENT\_FIRE =100535, /\* Fire alarm \*/

NETDEV\_DOOR\_EVENT\_CALL\_EMERGENCY =100536, /\* Emergency call alarm \*/

NETDEV\_DOOR\_PERSON\_FACE\_DOOR =100537, /\* Successful face recognition access \*/

NETDEV\_DOOR\_PERSON\_INVALE\_DOOR =100538, /\* Failed face recognition access \*/

NETDEV\_DOOR\_EVENT\_FINGERPRINT\_OPEN\_DOOR =100553, /\* Open door by fingerprint \*/

NETDEV\_DOOR\_EVENT\_ANTI\_SUBMARINE =100554, /\* Anti-passback alarm \*/

NETDEV\_DOOR\_EVENT\_PRESS\_MISTAKE =100555, /\* Press by mistake alarm \*/

NETDEV\_DOOR\_EVENT\_PF\_PW\_RF\_OPEN\_DOOR =100556, /\* Open door by fingerprint or password or card \*/

NETDEV\_DOOR\_EVENT\_DOOR\_FORCED\_OPEN\_CLEARED =100557, /\* Door forced open cleared \*/

NETDEV\_DOOR\_EVENT\_BLACKLIST =100558, /\* Blocklist \*/

NETDEV\_DOOR\_EVENT\_FAILED\_SEND\_CMD =100559, /\* Failed to send command \*/

NETDEV\_DOOR\_EVENT\_FAILED\_OPEN\_DOOE\_MU\_CARD =100560, /\*Failed to open door by multiple cards \*/

NETDEV\_DOOR\_EVENT\_CARD\_INVALID\_PERIOD =100561, /\*Swipe card during invalid period \*/

NETDEV\_DOOR\_EVENT\_CARD\_REPORT\_LOST =100562, /\*Card report lost \*/

NETDEV\_DOOR\_EVENT\_TAMPER\_ALARM =100563, /\* Tamper alarm \*/

NETDEV\_DOOR\_EVENT\_SWIPE\_CARD\_WHEN\_NORMAL\_OPEN =100564, /\* Swipe card when normally open \*/

NETDEV\_DOOR\_EVENT\_OPEN\_DOOR\_BY\_EMERGENCY\_PWD =100565, /\* Open door by emergency password \*/

NETDEV\_DOOR\_EVENT\_OPEN\_DOOR\_WHEN\_NORMAL\_OPEN =100566, /\* Open door when normally open \*/

NETDEV\_DOOR\_EVENT\_EXIT\_BUTTON\_DURING\_ILLEGAL\_PERIOD=100567,/\* Press exit button during illegal period \*/

NETDEV\_DOOR\_EVENT\_DOOR\_ALREADY\_OPEN =100568, /\* Door already opened \*/

NETDEV\_DOOR\_EVENT\_DOOR\_ALREADY\_CLOSE =100569, /\*Door already closed \*/

NETDEV\_DOOR\_EVENT\_ILLEGAL\_PERIOD =100570, /\* Illegal period \*/

NETDEV\_DOOR\_EVENT\_OPEN\_DOOR\_BY\_DURESS\_PWD =100571, /\* Open door by duress password \*/

NETDEV\_DOOR\_EVENT\_NO\_MASK =100572, /\*No mask \*/

NETDEV\_DOOR\_EVENT\_BODY\_TEMPERATURE =100573, /\*Abnormal body temperature \*/

NETDEV\_DOOR\_EVENT\_M0NITOR\_INPUT\_ACTIVATED = 100574, /\* Monitor input activated \*/

NETDEV\_DOOR\_EVENT\_M0NITOR\_INPUT\_CLEAR = 100575, /\* Monitor input cleared \*/

NETDEV\_DOOR\_EVENT\_END =100999, /\* End of definitions of access control events \*/

#define NETDEV\_IS\_DOOR\_EVENT(x) (100400 <= (x) && (x) < NETDEV\_DOOR\_EVENT\_END)

}NETDEV\_ALARM\_SUBTYPE\_E;

### Enumeration of alarm source types

typedef enum tagNETDEVAlarmSrcBelongType

{

NETDEV\_ALARM\_SRC\_BELONG\_TYPE\_OTHER = 0, /\* Other \*/

NETDEV\_ALARM\_SRC\_BELONG\_TYPE\_CHL = 1, /\* Channel alarm \*/

NETDEV\_ALARM\_SRC\_BELONG\_TYPE\_DEV = 2, /\* Device alarm \*/

NETDEV\_ALARM\_SRC\_BELONG\_TYPE\_INVALID = 0xffff /\* Invalid value \*/

}NETDEV\_ALARM\_SRC\_BELONG\_TYPE\_E;

### Enumeration of member genders

typedef enum tagNETDEVGenderType

{

NETDEV\_GENDER\_TYPE\_UNKNOW = 0, /\* 0-Unknown \*/

NETDEV\_GENDER\_TYPE\_MAN = 1, /\* 1-Male \*/

NETDEV\_GENDER\_TYPE\_WOMAN = 2, /\* 2-Female \*/

NETDEV\_GENDER\_TYPE\_UNEXPLAINED = 9, /\* 9-Unexplained \*/

NETDEV\_GENDER\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_GENDER\_TYPE\_E;

### Enumeration of ID types

typedef enum tagNETDEVCertificateType

{

NETDEV\_CERTIFICATE\_TYPE\_ID = 0, /\*0: ID card \*/

NETDEV\_CERTIFICATE\_TYPE\_IC = 1, /\* 1: IC card \*/

NETDEV\_CERTIFICATE\_TYPE\_PASSPORT = 2, /\* 2: Passport \*/

NETDEV\_CERTIFICATE\_TYPE\_DRIVING\_LICENSE = 3, /\* 3: Driver’s license \*/

NETDEV\_CERTIFICATE\_TYPE\_OTHER = 99, /\* 99: Other \*/

NETDEV\_CERTIFICATE\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_ID\_TYPE\_E;

### Enumeration of file info

typedef enum tagNETDEVFileType

{

NETDEV\_TYPE\_FOLDER = 0, /\* Folder \*/

NETDEV\_TYPE\_FILE = 1, /\* File \*/

NETDEV\_TYPE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_FILE\_TYPE\_E;

### Enumeration of face processing result codes

typedef enum tagNETDEVPersonResultCode

{

NETDEV\_PERSON\_CODE\_SUCCEED = 0, /\* Success \*/

NETDEV\_PERSON\_CODE\_COMMON\_FAIL = 1, /\* Common failure \*/

NETDEV\_PERSON\_CODE\_SENDING = 2, /\* Sending \*/

NETDEV\_PERSON\_CODE\_DEV\_NOT\_SUPPORT = 4, /\* Not supported by device \*/

NETDEV\_PERSON\_CODE\_ARGORITHM\_INIT\_FAIL = 1000, /\* Algorithm initialization failed \*/

NETDEV\_PERSON\_CODE\_FACE\_DETECT\_FAIL = 1001, /\* Face detection failed \*/

NETDEV\_PERSON\_CODE\_PICTURE\_NO\_FACE = 1002, /\* No face in image \*/

NETDEV\_PERSON\_CODE\_JPEG\_DECODE\_FAIL = 1003, /\* Failed to decode JPEG image \*/

NETDEV\_PERSON\_CODE\_PICTURE\_QUALITY\_LOW = 1004, /\* Low image quality \*/

NETDEV\_PERSON\_CODE\_PICTURE\_ZOOM\_FAIL = 1005, /\* Failed to zoom image \*/

NETDEV\_PERSON\_CODE\_INTELLECT\_DISABLE = 1006, /\* Intelligent function disabled \*/

NETDEV\_PERSON\_CODE\_PICTURE\_TOO\_SMALL = 1007, /\* Image is too small \*/

NETDEV\_PERSON\_CODE\_PICTURE\_TOO\_LARGE = 1008, /\* Image is too large \*/

NETDEV\_PERSON\_CODE\_RESOLUTION\_TOO\_LARGE = 1009, /\* Image exceeds 1920\*1080 \*/

NETDEV\_PERSON\_CODE\_PICTURE\_NON\_EXISTENT = 1010, /\* Image does not exist \*/

NETDEV\_PERSON\_CODE\_FACE\_ELEMENTS\_LIMIT = 1011, /\* Number of face elements reached upper limit \*/

NETDEV\_PERSON\_CODE\_INTELLECT\_MODULE\_MISMATCH = 1012, /\* Smart bar version does not match the device \*/

NETDEV\_PERSON\_CODE\_DOCUMENT\_ID\_INVLID = 1013, /\* Invalid ID number of imported library member \*/

NETDEV\_PERSON\_CODE\_PICTURE\_FORMAT\_ERROR = 1014, /\* Invalid image format of imported library member \*/

NETDEV\_PERSON\_CODE\_MONITOR\_DEVICE\_LIMIT = 1015, /\* Channel’s monitoring capability reached upper limit \*/

NETDEV\_PERSON\_CODE\_FACE\_LIBRARY\_LOCKED = 1016, /\* Face library is being operated by another client \*/

NETDEV\_PERSON\_CODE\_FACE\_LIBRARY\_UPDATING = 1017, /\* Face library file is being updated \*/

NETDEV\_PERSON\_CODE\_JSON\_DESERIALIZE\_FAIL = 1018, /\* Json deserialization failed \*/

NETDEV\_PERSON\_CODE\_BASE64\_DECODE\_FAIL = 1019, /\* Base64 decoding failed \*/

NETDEV\_PERSON\_CODE\_PICTURE\_SIZE\_MISMATCH = 1020, /\* Size of encoded face image does not match the received \*/

NETDEV\_PERSON\_CODE\_DEV\_PROTOCOL\_DIFFER = 1021, /\* A monitoring task must use devices connected via the same image protocol \*/

NETDEV\_PERSON\_CODE\_REACH\_LIMIT = 1022, /\* Device’s face libraries reached upper limit \*/

NETDEV\_PERSON\_CODE\_NO\_SMART = 1023, /\* No smart bar \*/

NETDEV\_PERSON\_CODE\_DEV\_BUSY = 1024, /\* Device is busy \*/

NETDEV\_PERSON\_CODE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_PERSON\_RESULT\_CODE\_E;

### Enumeration of person comparison results

typedef enum tagPersonCompareResult

{

NETDEV\_TYPE\_COMPARE\_SUCCESS = 1, /\* Comparison succeeded \*/

NETDEV\_TYPE\_COMPARE\_FAILED = 2, /\* Comparison failed \*/

NETDEV\_TYPE\_COMPARE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_PERSON\_COMPARE\_RESULT\_TYPE\_E;

### Enumeration of linkage actions

typedef enum tagNETDEVAlarmActID

{

ALARM\_ACTION\_TYPE\_NVR\_PREVIEW = 0, /\* Preview on NVR. See NETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_BUZZER = 1, /\* Buzzer. Not supported by IPC. For NVR, see NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_E\_MAIL = 2, /\* E-Mail. Not supported by IPC. For NVR see NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_RECORD = 3, /\* Recording. Not supported by IPC. For NVR, see NETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_PRESET = 4, /\* PTZ preset. See NETDEV\_PRESET\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_OUTPUT\_SWITCH = 5, /\* Alarm output. See NETDEV\_OUTPUT\_SWITCH\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_SNAP = 6, /\* Snapshot. For IPC, ActParam is not required. For NVR, see NETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S\*/

ALARM\_ACTION\_TYPE\_BOX = 7, /\* Pop-up alarm window. Not supported by IPC. For NVR, see NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_CENTER\_RECORD = 8, /\* Central recording. For IPC, ActParam is not required. Not supported by NVR. \*/

ALARM\_ACTION\_TYPE\_START\_LOCAL\_RECORD = 9, /\* Start local recording. For IPC, ActParam is not required. Not supported by NVR. \*/

ALARM\_ACTION\_TYPE\_STOP\_LOCAL\_RECORD = 10, /\* Stop local recording. For IPC, ActParam is not required. Not supported by NVR. \*/

ALARM\_ACTION\_TYPE\_SNAP\_UP\_FTP = 11, /\* Upload snapshot image via FTP. For IPC, ActParam is not required. Not supported by NVR. \*/

ALARM\_ACTION\_TYPE\_SNAP\_UP\_EMAIL = 12, /\* Send snapshot image by email. For IPC, ActParam is not required. Not supported by NVR. \*/

ALARM\_ACTION\_TYPE\_SNAP\_UP\_FTP\_AND\_EMAIL = 13, /\* Upload/send snapshot image by FTP/email. For IPC, ActParam is not required. Not supported by NVR. \*/

ALARM\_ACTION\_TYPE\_SMART\_SNAP\_UP = 14, /\* Upload smart snapshot image. For IPC, ActParam is not required. Not supported by NVR. \*/

ALARM\_ACTION\_TYPE\_FACE\_PIC\_SNAP\_UP = 15, /\* Capture and upload face image. For IPC, ActParam is not required. Not supported by NVR. \*/

ALARM\_ACTION\_TYPE\_ALARM\_REPORT = 16, /\* Alarm report. For IPC, ActParam is not required. Not supported by NVR. \*/

ALARM\_ACTION\_TYPE\_PTZ\_ZOOM = 17, /\* PTZ zoom. Not supported by IPC. \*/

ALARM\_ACTION\_TYPE\_INVALID = 0xff /\* Invalid parameter \*/

}NETDEV\_ALARM\_ACT\_ID\_E;

### Enumeration alarm output status

typedef enum tagNETDEVRelayOutPutState

{

NETDEV\_BOOLEAN\_STATUS\_ACTIVE = 0, /\* Active \*/

NETDEV\_BOOLEAN\_STATUS\_INACTIVE = 1 /\* Inactive \*/

}NETDEV\_RELAYOUTPUT\_STATE\_E;

### Enumeration of arming schedule enablement

typedef enum tagNETDEVDefencePlanEnable

{

NETDEV\_DEFENCE\_PLAN\_DISABLE = 0, /\* Disable \*/

NETDEV\_DEFENCE\_PLAN\_ENABLE = 1 /\* Enable \*/

}NETDEV\_DEFENCE\_PLAN\_ENABLE\_E;

### Enumeration of days of a week

typedef enum tagNETDEVWeekInfo

{

NETDEV\_LAPI\_WEEK\_MONDAY = 1, /\* Monday \*/

NETDEV\_LAPI\_WEEK\_TUESDAY = 2, /\* Tuesday \*/

NETDEV\_LAPI\_WEEK\_WEDNESDAY = 3, /\* Wednesday \*/

NETDEV\_LAPI\_WEEK\_THURSDAY = 4, /\* Thursday \*/

NETDEV\_LAPI\_WEEK\_FRIDAY = 5, /\* Friday \*/

NETDEV\_LAPI\_WEEK\_SATURDAY = 6, /\* Saturday \*/

NETDEV\_LAPI\_WEEK\_SUNDAY = 7, /\* Sunday \*/

NETDEV\_LAPI\_WEEK\_HOLIDAY = 8, /\* Holiday \*/

NETDEV\_LAPI\_WEEK\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_LAPI\_WEEK\_INFO\_E;

### Enumeration of arming types

typedef enum tagNETDEVArmingType

{

NETDEV\_ARMING\_TYPE\_TIMING = 0, /\* Schedule \*/

NETDEV\_ARMING\_TYPE\_MOTIONDETEC = 1, /\* Motion detection \*/

NETDEV\_ARMING\_TYPE\_ALARM = 2, /\* Alarm \*/

NETDEV\_ARMING\_TYPE\_MOTIONDETEC\_AND\_ALARM = 3, /\* Motion detection and alarm \*/

NETDEV\_ARMING\_TYPE\_MOTIONDETEC\_OR\_ALARM = 4, /\* Motion detection or alarm \*/

NETDEV\_ARMING\_TYPE\_NO\_PLAN = 5, /\* No schedule \*/

NETDEV\_ARMING\_TYPE\_EVENT = 10 /\* Event \*/

}NETDEV\_ARMING\_TYPE\_E;

**Remarks:**

* Recording and snapshot schedules support all arming types; other schedule types only support Normal. IPC only supports Normal (default is Normal, and this field is unnecessary).

### Enumeration of person library types

typedef enum tagNETDEVPeopleLibType

{

NETDEV\_PEOPLE\_LIB\_TYPE\_DEFAULT = 0, /\* Default (null) \*/

NETDEV\_PEOPLE\_LIB\_TYPE\_BLACKLIST = 1, /\* Blocklist \*/

NETDEV\_PEOPLE\_LIB\_TYPE\_STRANGER = 2, /\* Greylist/stranger \*/

NETDEV\_PEOPLE\_LIB\_TYPE\_STAFF = 3, /\* Staff \*/

NETDEV\_PEOPLE\_LIB\_TYPE\_VISITOR = 4, /\* Visitor \*/

NETDEV\_PEOPLE\_LIB\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_PEOPLE\_LIB\_TYPE\_E;

### Enumeration of face monitoring operation result codes

typedef enum tagNETDEVPersonMonitorOptResCode

{

NETDEV\_PERSON\_MONITOR\_CODE\_INIT\_DETECT\_FAIL = 11702, /\* Failed to initialize detection \*/

NETDEV\_PERSON\_MONITOR\_CODE\_FACE\_DETECT\_FAIL = 11703, /\* Face detection failed \*/

NETDEV\_PERSON\_MONITOR\_CODE\_IMAGE\_NOT\_FIND\_FACE = 11704, /\* No face in image \*/

NETDEV\_PERSON\_MONITOR\_CODE\_JPEG\_PARSE\_FAIL = 11705, /\* Failed to decode JPEG image \*/

NETDEV\_PERSON\_MONITOR\_CODE\_IMAGE\_MASS\_NOT\_ENOUGH = 11706, /\* Low face image quality \*/

NETDEV\_PERSON\_MONITOR\_CODE\_IMAGE\_ZOOM\_FAIL = 11707, /\* Failed to zoom image \*/

NETDEV\_PERSON\_MONITOR\_CODE\_NOT\_START\_SMART = 11708, /\* Intelligent function disabled \*/

NETDEV\_PERSON\_MONITOR\_CODE\_PICTURE\_TOO\_SMALL = 11709, /\* Image is too small \*/

NETDEV\_PERSON\_MONITOR\_CODE\_CREATE\_FACE\_LIB\_FAIL = 11710, /\* Failed to create face library \*/

NETDEV\_PERSON\_MONITOR\_CODE\_CREATE\_MONITOR\_FAIL = 11711, /\* Failed to create monitoring task \*/

NETDEV\_PERSON\_MONITOR\_CODE\_PICTURE\_TOO\_LARGE = 11714, /\* Image is too large \*/

NETDEV\_PERSON\_MONITOR\_CODE\_RESOLUTION\_TOO\_LARGE = 11715, /\* Image exceeds 1920\*1080 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_PICTURE\_NON\_EXISTENT = 11716, /\* Image does not exist \*/

NETDEV\_PERSON\_MONITOR\_CODE\_FACE\_ELEMENTS\_LIMIT = 11717, /\* Number of face elements reached upper limit \*/

NETDEV\_PERSON\_MONITOR\_CODE\_INTELLECT\_MODULE\_MISMATCH = 11718, /\* Smart bar version does not match the device \*/

NETDEV\_PERSON\_MONITOR\_CODE\_DOCUMENT\_ID\_INVLID = 11719, /\* Invalid ID number of imported library member \*/

NETDEV\_PERSON\_MONITOR\_CODE\_PICTURE\_FORMAT\_ERROR = 11720, /\* Invalid image format of imported library member \*/

NETDEV\_PERSON\_MONITOR\_CODE\_MONITOR\_DEVICE\_LIMIT = 11721, /\* Channel’s monitoring capability reached upper limit \*/

NETDEV\_PERSON\_MONITOR\_CODE\_FACE\_LIBRARY\_LOCKED = 11722, /\* Face library is being operated by another client \*/

NETDEV\_PERSON\_MONITOR\_CODE\_FACE\_LIBRARY\_UPDATING = 11723, /\* Face library file is being updated \*/

NETDEV\_PERSON\_MONITOR\_CODE\_JSON\_DESERIALIZE\_FAIL = 11724, /\* Json deserialization failed \*/

NETDEV\_PERSON\_MONITOR\_CODE\_BASE64\_DECODE\_FAIL = 11725, /\* Base64 decoding failed \*/

NETDEV\_PERSON\_MONITOR\_CODE\_PICTURE\_SIZE\_MISMATCH = 11726 /\* Size of encoded face image does not match the received \*/

}NETDEV\_PERSON\_MONITOR\_OPT\_RES\_CODE\_E;

### Enumeration of alarm snapshot types

typedef enum tagNETDEVAlarmSnapShotType

{

NETDEV\_ALARM\_SNAPSHOT\_MOTION\_DETECT = 0, /\* Motion detection alarm \*/

NETDEV\_ALARM\_SNAPSHOT\_VIDEO\_LOST = 1, /\* Video loss alarm \*/

NETDEV\_ALARM\_SNAPSHOT\_AUDIO\_DETECTION = 2, /\* Audio detection alarm \*/

NETDEV\_ALARM\_SNAPSHOT\_INPUT\_SWITCH = 3, /\* Input alarm \*/

NETDEV\_ALARM\_SNAPSHOT\_IPC\_OFFLINE = 4, /\* IPC offline \*/

NETDEV\_ALARM\_SNAPSHOT\_TAMPER\_DETECT = 5, /\* Tampering detection \*/

NETDEV\_ALARM\_SNAPSHOT\_CROSS\_LINE = 6, /\* Cross line detection, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_INTROSION\_ZONE = 7, /\* Intrusion detection, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_FACE\_DETECT = 8, /\* Face detection, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_SCENCECHANGE = 9, /\* Scene change, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_OUT\_FOCUS = 10, /\* Defocus detection, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_FACE\_MATCH = 11, /\* Face match alarm, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_FACE\_NOT\_MATCH = 12, /\* Face not match alarm, supported by VSM only \*/

NETDEV\_ALARM\_SNAPSHOT\_GATHER = 13, /\* People gathering, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_REMAIN\_ARTICLE = 14, /\* Object left behind, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_SMART\_TRACK = 15, /\* Auto track, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_LOITERING\_DETECTOR = 16, /\* Loitering detection, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_VEHICLE\_MATCH = 17, /\* Vehicle match alarm, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_VEHICLE\_NOT\_MATCH = 18, /\* Vehicle not match alarm, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_ENTER\_AREA = 19, /\* Enter area, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_LEAVE\_AREA = 20, /\* Leave area, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_ARTICLE\_MOVE = 21, /\* Object removed, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_DURESS = 22, /\* Duress alarm, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_ANRIDISASSEMBLY = 23, /\* Anti-dismantle alarm, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_ANRIDISASSEMBLY\_RECOVER = 24, /\* Anti-dismantle alarm, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_BYPASS\_OPERATION = 25, /\* Bypass operation, supported by VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_CF\_BJ\_EMBNAM\_ZONE = 26, /\* EMB zone alarm. Applicable to VMS only \*/

NETDEV\_ALARM\_SNAPSHOT\_CONFLAGRATION = 27, /\* Fire detection alarm \*/

NETDEV\_ALARM\_SNAPSHOT\_HUMAN\_SHAPE\_DETECT = 28, /\* Human body detection alarm \*/

NETDEV\_ALARM\_SNAPSHOT\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_ALARM\_SNAPSHOT\_TYPE\_E;

### Enumeration of 5ePTZ display modes

typedef enum tagNETDEVPTZDisplayMode

{

NETDEV\_PTZ\_DISPLAY\_ORIGINAL = 0, /\*Original image \*/

NETDEV\_PTZ\_DISPLAY\_5PTZ = 1, /\*Panoramic+5ePTZ\*/

NETDEV\_PTZ\_DISPLAY\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_PTZ\_DISPLAY\_MODE\_E;

### Enumeration of fisheye dewarping modes

typedef enum tagNetDEVFishEyePtzMode

{

NETDEV\_FISHEYE\_MODE\_ORIGINAL = 0, /\* Original image \*/

NETDEV\_FISHEYE\_MODE\_180 = 1, /\* 2\*180°\*/

NETDEV\_FISHEYE\_MODE\_360\_1PTZ = 2, /\* 360°+1PTZ \*/

NETDEV\_FISHEYE\_MODE\_360\_6PTZ = 3, /\* 360°+6PTZ \*/

NETDEV\_FISHEYE\_MODE\_3PTZ = 4, /\* Fisheye+3PTZ\*/

NETDEV\_FISHEYE\_MODE\_MID\_ON\_4PTZ = 5, /\* Fisheye in the middle and display+4PTZ \*/

NETDEV\_FISHEYE\_MODE\_MID\_OFF\_4PTZ= 6, /\* Fisheye in the middle but not display+4PTZ \*/

NETDEV\_FISHEYE\_MODE\_LEFT\_4PTZ = 7, /\* Fisheye on the left+4PTZ \*/

NETDEV\_FISHEYE\_MODE\_8PTZ = 8, /\* Fisheye+8PTZ \*/

NETDEV\_FISHEYE\_MODE\_PANORAMA = 9, /\* Panoramic \*/

NETDEV\_FISHEYE\_MODE\_PR\_3PTZ = 10, /\* Panoramic+3PTZ \*/

NETDEV\_FISHEYE\_MODE\_PR\_4PTZ = 11, /\* Panoramic+4PTZ \*/

NETDEV\_FISHEYE\_MODE\_PR\_8PTZ = 12, /\* Panoramic+8PTZ \*/

NETDEV\_FISHEYE\_MODE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_FISHEYE\_PTZ\_MODE\_E;

### Enumeration of device mounting modes

typedef enum tagNETDEVfInstallMode

{

NETDEV\_INSTALL\_MODE\_TOP = 0, /\* Top mount\*/

NETDEV\_INSTALL\_MODE\_BOTTOM = 1, /\* Bottom mount \*/

NETDEV\_INSTALL\_MODE\_SIDE = 2, /\* Side mount \*/

NETDEV\_INSTALL\_MODE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_INSTALL\_MODE\_E;

### Enumeration of mouse movement modes

typedef enum tagNETDEVfMouseMoveMode

{

NETDEV\_MOUSE\_MOVE = 0, /\* Mouse move \*/

NETDEV\_MOUSE\_LEFT\_BTN\_DOWN = 1, /\* Left button down \*/

NETDEV\_MOUSE\_LEFT\_BTN\_UP = 2, /\* Left button up \*/

NETDEV\_MOUSE\_WHEEL = 3, /\* Scroll wheel \*/

NETDEV\_MOUSE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_MOUSE\_MOVE\_MODE\_E;

### Enumeration of recording types

typedef enum tagNETDEVRecordType

{

NETDEV\_RECORD\_TYPE\_MANUAL = 0, /\* Manual recording \*/

NETDEV\_RECORD\_TYPE\_NORMAL = 1 /\* Normal recording \*/

}NETDEV\_RECORD\_TYPE\_E;

### Enumeration of recording status

typedef enum tagNETDEVManualRecordStatus

{

NETDEV\_MANUAL\_RECORD\_STATUS\_FREE = 0, /\* Not recording \*/

NETDEV\_MANUAL\_RECORD\_STATUS\_RECORDING = 1, /\* Recording \*/

NETDEV\_MANUAL\_RECORD\_STATUS\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_MANUAL\_RECORD\_STATUS\_E;

### Device configuration commands

typedef enum tagNETDEVCfgCmd

{

NETDEV\_GET\_DEVICECFG = 100, /\* Get device info.

See [NETDEV\_DEVICE\_BASICINFO\_S](#_设备基本信息) \*/

NETDEV\_SET\_DEVICECFG = 101, /\* Reserved \*/

NETDEV\_GET\_NTPCFG = 110, /\* Get NTP parameters.

See [NETDEV\_SYSTEM\_NTP\_INFO\_S](#_NTP参数) \*/

NETDEV\_SET\_NTPCFG = 111, /\* Set NTP parameters.

See [NETDEV\_SYSTEM\_NTP\_INFO\_S](#_NTP参数) \*/

NETDEV\_GET\_NTPCFG\_EX = 112, /\* Get NTP parameters (extended, recommended).

See [NETDEV\_SYSTEM\_NTP\_INFO\_LIST\_S](#_NTP列表) \*/

NETDEV\_SET\_NTPCFG\_EX = 113, /\* Set NTP parameters (extended, recommended).

See [NETDEV\_SYSTEM\_NTP\_INFO\_LIST\_S](#_NTP列表) \*/

NETDEV\_GET\_STREAMCFG = 120, /\* Get video encoding parameters.

See [NETDEV\_VIDEO\_STREAM\_INFO\_S](#_通道视频流信息) \*/

NETDEV\_SET\_STREAMCFG = 121, /\* Set video encoding parameters.

See [NETDEV\_VIDEO\_STREAM\_INFO\_S](#_通道视频流信息) \*/

NETDEV\_GET\_STREAMCFG\_EX = 122, /\* Get video encoding parameters (extended, recommended).

See [NETDEV\_VIDEO\_STREAM\_INFO\_LIST\_S](#_视频流信息列表) \*/

NETDEV\_SET\_STREAMCFG\_EX = 123, /\* Set video encoding parameters (extended, recommended).

See [NETDEV\_VIDEO\_STREAM\_INFO\_LIST\_S](#_视频流信息列表) \*/

NETDEV\_GET\_VIDEOMODECFG = 124, /\* Get image collection modes.

See [NETDEV\_VIDEO\_MODE\_INFO\_S](#_视频制式能力) \*/

NETDEV\_SET\_VIDEOMODECFG = 125, /\* Set image collection modes.

See [NETDEV\_VIDEO\_MODE\_INFO\_S](#_视频制式能力) \*/

NETDEV\_GET\_OSDCFG = 140, /\* Get OSD configuration info.

See [NETDEV\_VIDEO\_OSD\_CFG\_S](#_通道OSD的基本属性信息) \*/

NETDEV\_SET\_OSDCFG = 141, /\* Set OSD configuration info.

See [NETDEV\_VIDEO\_OSD\_CFG\_S](#_通道OSD的基本属性信息) \*/

NETDEV\_GET\_OSDCFG\_EX = 142, /\* Get OSD configuration list.

See [NETDEV\_CHL\_OSD\_CFG\_LIST\_S](#_通道OSD配置列表) \*/

NETDEV\_SET\_OSDCFG\_EX = 143, /\* Set OSD configuration list.

See [NETDEV\_CHL\_OSD\_CFG\_LIST\_S](#_通道OSD配置列表) \*/

NETDEV\_GET\_OSD\_CONTENT\_CFG = 144, /\* Get OSD configuration info (extended, recommended).

See [NETDEV\_OSD\_CONTENT\_S](#_通道OSD所有内容) \*/

NETDEV\_SET\_OSD\_CONTENT\_CFG = 145, /\* Set OSD configuration info (extended, recommended).

See [NETDEV\_OSD\_CONTENT\_S](#_通道OSD所有内容) \*/

NETDEV\_GET\_OSD\_CONTENT\_STYLE\_CFG = 146, /\* Get OSD content sytle.

See [NETDEV\_OSD\_CONTENT\_STYLE\_S](#_通道OSD内容样式) \*/

NETDEV\_SET\_OSD\_CONTENT\_STYLE\_CFG = 147, /\* Set OSD content style.

See [NETDEV\_OSD\_CONTENT\_STYLE\_S](#_通道OSD内容样式) \*/

NETDEV\_GET\_ALARM\_OUTPUTCFG = 150, /\* Get alarm output configuration info.

See [NETDEV\_ALARM\_OUTPUT\_LIST\_S](#_所有告警开关量输入信息) \*/

NETDEV\_SET\_ALARM\_OUTPUTCFG = 151, /\* Set alarm output configuration info.

See [NETDEV\_ALARM\_OUTPUT\_LIST\_S](#_所有告警开关量输入信息) \*/

NETDEV\_TRIGGER\_ALARM\_OUTPUT = 152, /\* Trigger alarm output.

See [NETDEV\_TRIGGER\_ALARM\_OUTPUT\_S](#_触发开关量输入告警) \*/

NETDEV\_GET\_ALARM\_INPUTCFG = 153, /\* Get the number of alarm inputs.

See [NETDEV\_ALARM\_INPUT\_LIST\_S](#_所有告警开关量输入信息_1) \*/

NETDEV\_GET\_MANUAL\_ALARM\_CFG = 154, /\* Get manual alarm configuration info.

See [NETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_LIST\_S](#_输出开关量的逻辑报警状态列表(手动告警)) \*/

NETDEV\_SET\_MANUAL\_ALARM\_CFG = 155, /\* Set manual alarm configuration info.

See [NETDEV\_OUTPUT\_SWITCH\_MANUAL\_ALARM\_INFO\_S](#_触发或清除输出开关量的逻辑报警状态结构体(手动告警)) \*/

NETDEV\_GET\_INPUTSWITCH\_LIST\_CFG = 156, /\* Get all alarm input configuration.

See [NETDEV\_INPUT\_SWITCH\_INFO\_LIST\_S](#_输入开关量告警配置信息列表) \*/

NETDEV\_GET\_IMAGECFG = 160, /\* Get image configuration info.

See [NETDEV\_IMAGE\_SETTING\_S](#_设备图像设置) \*/

NETDEV\_SET\_IMAGECFG = 161, /\* Set image configuration info.

See [NETDEV\_IMAGE\_SETTING\_S](#_设备图像设置) \*/

NETDEV\_GET\_IMAGE\_EXPOSURE = 162, /\* Get image exposure parameters.

See [NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)\*/

NETDEV\_SET\_IMAGE\_EXPOSURE = 163, /\* Set image exposure parameters.

See [NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数) \*/

NETDEV\_GET\_IMAGE\_LAMP\_CTRLCFG = 164, /\* Get illuminator parameters.

See [NETDEV\_LAMP\_CTRL\_INFO\_S](#_补光灯信息) \*/

NETDEV\_SET\_IMAGE\_LAMP\_CTRLCFG = 165, /\* Set illuminator parameters.

See [NETDEV\_LAMP\_CTRL\_INFO\_S](#_补光灯信息) \*/

NETDEV\_GET\_IMAGE\_WHITE\_BALANCE = 166, /\* Get image white balance parameters.

See  [NETDEV\_WHITE\_BALANCE\_INFO\_S](#_白平衡信息) \*/

NETDEV\_SET\_IMAGE\_WHITE\_BALANCE = 167, /\* Set image white balance parameters.

See  [NETDEV\_WHITE\_BALANCE\_INFO\_S](#_白平衡信息) \*/

NETDEV\_SET\_IMAGE\_PARAM\_RESET = 168, /\* Set default image parameters. \*/

NETDEV\_GET\_NETWORKCFG = 170, /\* Get network configuration info.

See [NETDEV\_NETWORKCFG\_S](#_网络配置信息) \*/

NETDEV\_SET\_NETWORKCFG = 171, /\* Set network configuration info.

See [NETDEV\_NETWORKCFG\_S](#_网络配置信息) \*/

NETDEV\_GET\_VPN\_CFG = 172, /\* Get VPN client’s specified mode.

See [NETDEV\_VPN\_CLIENT\_INFO\_S](#_VPN客户端指定模式信息) \*/

NETDEV\_SET\_VPN\_CFG = 173, /\* Set VPN client's specified mode.

See [NETDEV\_VPN\_CLIENT\_INFO\_S](#_VPN客户端指定模式信息) \*/

NETDEV\_SET\_DNS\_CFG = 174, /\* Set DNS address info.

See [NETDEV\_DNS\_INFO\_S](#_设备DNS信息) \*/

NETDEV\_GET\_DNS\_CFG = 175, /\* Get DNS address info.

See [NETDEV\_DNS\_INFO\_S](#_设备DNS信息) \*/

NETDEV\_SET\_NETWORK\_CARDS = 176, /\* Set configuration info of all NICs.

See [NETDEV\_NETWORK\_CARD\_INFO\_S](#_网卡信息) \*/

NETDEV\_GET\_NETWORK\_CARDS = 177, /\* Get configuration info of all NICs.

See [NETDEV\_NETWORK\_CARD\_INFO\_S](#_网卡信息) \*/

NETDEV\_SET\_NETWORK\_PORTS = 178, /\* Set network port info.

See [NETDEV\_NETWORK\_PORTS\_INFO\_S](#_端口信息) \*/

NETDEV\_GET\_NETWORK\_PORTS = 179, /\* Get network port info.

See [NETDEV\_NETWORK\_PORTS\_INFO\_S](#_端口信息) \*/

NETDEV\_GET\_PRIVACYMASKCFG = 180, /\* Get privacy mask configuration info.

See [NETDEV\_PRIVACY\_MASK\_CFG\_S](#_隐私遮盖配置信息) \*/

NETDEV\_SET\_PRIVACYMASKCFG = 181, /\* Set privacy mask configuration info.

See [NETDEV\_PRIVACY\_MASK\_CFG\_S](#_隐私遮盖配置信息) \*/

NETDEV\_DELETE\_PRIVACYMASKCFG = 182, /\* Delete privacy mask configuration info. \*/

NETDEV\_GET\_TAMPERALARM = 190, /\* Get tampering detection alarm info.

See [NETDEV\_TAMPER\_ALARM\_INFO\_S](#_遮挡检测分析信息) \*/

NETDEV\_SET\_TAMPERALARM = 191, /\* Set tampering detection alarm info.

See [NETDEV\_TAMPER\_ALARM\_INFO\_S](#_遮挡检测分析信息) \*/

NETDEV\_GET\_MOTIONALARM = 200, /\* Get motion detection alarm info.

See [NETDEV\_MOTION\_ALARM\_INFO\_S](#_运动检测分析信息) \*/

NETDEV\_SET\_MOTIONALARM = 201, /\* Set motion detection alarm info.

See [NETDEV\_MOTION\_ALARM\_INFO\_S](#_运动检测分析信息) \*/

NETDEV\_GET\_CROSSLINEALARM = 202, /\* Get cross line detection alarm info.

See [NETDEV\_CROSS\_LINE\_ALARM\_INFO\_S](#_越界检测分析信息) \*/

NETDEV\_SET\_CROSSLINEALARM = 203, /\* Set cross line detection alarm info.

See [NETDEV\_CROSS\_LINE\_ALARM\_INFO\_S](#_越界检测分析信息) \*/

NETDEV\_GET\_INTRUSIONALARM = 204, /\* Get intrusion detection alarm info.

See [NETDEV\_INTRUSION\_ALARM\_INFO\_S](#_入侵检测分析信息) \*/

NETDEV\_SET\_INTRUSIONALARM = 205, /\* Set intrusion detection alarm info.

See [NETDEV\_INTRUSION\_ALARM\_INFO\_S](#_入侵检测分析信息) \*/

NETDEV\_GET\_DISKSINFO = 210, /\* Get HDD info.

See [NETDEV\_DISK\_INFO\_LIST\_S](#_硬盘信息列表) \*/

NETDEV\_SET\_DISKSINFO = 211, /\* Set HDD info. Reserved \*/

NETDEV\_GET\_FOCUSINFO = 230, /\* Get focus info.

See [NETDEV\_FOCUS\_INFO\_S](#_聚焦信息) \*/

NETDEV\_SET\_FOCUSINFO = 231, /\* Set focus info.

See [NETDEV\_FOCUS\_INFO\_S](#_聚焦信息) \*/

NETDEV\_GET\_IRCUTFILTERINFO = 232, /\* Get day/night mode info.

See [NETDEV\_IRCUT\_FILTER\_INFO\_S](#_昼夜模式信息_1) \*/

NETDEV\_SET\_IRCUTFILTERINFO = 233, /\* Set day/night mode info.

See [NETDEV\_IRCUT\_FILTER\_INFO\_S](#_昼夜模式信息_1) \*/

NETDEV\_GET\_DEFOGGINGINFO = 234, /\* Get defogging mode info.

See [NETDEV\_DEFOGGING\_INFO\_S](#_透雾信息) \*/

NETDEV\_SET\_DEFOGGINGINFO = 235, /\* Set defogging mode info.

See [NETDEV\_DEFOGGING\_INFO\_S](#_透雾信息) \*/

NETDEV\_SET\_CHL\_IMAGE\_ENHANCE = 236, /\* Set image enhancement parameters.

See [NETDEV\_IMAGE\_ENHANCE\_S](#_指定通道的图像增强参数) \*/

NETDEV\_GET\_CHL\_IMAGE\_ENHANCE = 237, /\* Get image enhancement parameters.

See [NETDEV\_IMAGE\_ENHANCE\_S](#_指定通道的图像增强参数) \*/

NETDEV\_GET\_AUDIOIN\_CFG = 240, /\* Get audio input configuration info.

See [NETDEV\_AUDIO\_INPUT\_CFG\_INFO\_S](#_音频输入参数配置信息) \*/

NETDEV\_SET\_AUDIOIN\_CFG = 241, /\* Set audio input configuration info.

See [NETDEV\_AUDIO\_INPUT\_CFG\_INFO\_S](#_音频输入参数配置信息) \*/

NETDEV\_GET\_DST\_CFG = 260, /\* Get DST configuration info.

See [NETDEV\_DST\_CFG\_S](#_夏令时配置信息) \*/

NETDEV\_SET\_DST\_CFG = 261, /\* Set DST configuration info.

See [NETDEV\_DST\_CFG\_S](#_夏令时配置信息) \*/

NETDEV\_GET\_RECORDPLANINFO = 270, /\* Get recording schedule configuration info. Supported by NVR and VMS. See [NETDEV\_RECORD\_PLAN\_CFG\_INFO\_S](#_录像计划配置信息)\*/

NETDEV\_SET\_RECORDPLANINFO = 271, /\* Set recording schedule configuration info. Supported by NVR and VMS. See [NETDEV\_RECORD\_PLAN\_CFG\_INFO\_S](#_录像计划配置信息) \*/

NETDEV\_SET\_SNAPSHOT\_CFG = 272, /\* Set snapshot parameters.

See [NETDEV\_VIDEO\_SNAPSHOT\_S](#_视频输入通道的抓图信息) \*/

NETDEV\_GET\_SNAPSHOT\_CFG = 273, /\* Get snapshot parameters.

See [NETDEV\_VIDEO\_SNAPSHOT\_S](#_视频输入通道的抓图信息)\*/

NETDEV\_GET\_MOTIONDETECTION\_WEEKPLAN = 280, /\* Get motion detection arming schedule.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_MOTIONDETECTION\_WEEKPLAN = 281, /\* Set motion detection arming schedule.

Set [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_CROSSLINEDETECTION\_WEEKPLAN = 282, /\* Get cross line detection arming schedule.

Set [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_CROSSLINEDETECTION\_WEEKPLAN = 283, /\* Set cross line detection arming schedule.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_INTRUSIONDETECTION\_WEEKPLAN = 284, /\* Get intrusion detection arming schedule.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_INTRUSIONDETECTION\_WEEKPLAN = 285, /\* Set intrusion detection arming schedule.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_AUDIODETECTION\_WEEKPLAN = 286, /\* Get audio detection arming schedule.

Set [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_AUDIODETECTION\_WEEKPLAN = 287, /\* Set audio detection arming schedule.

Set [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_OBJTRACK\_WEEKPLAN = 288, /\* Get auto track arming scheduling.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_OBJTRACK\_WEEKPLAN = 289, /\* Set auto track arming scheduling.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_VIDEOLOSS\_WEEKPLAN = 290, /\* Get video loss arming schedule.

Set [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_VIDEOLOSS\_WEEKPLAN = 291, /\* Set video loss arming schedule.

Set [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_TAMPERDETECTION\_WEEKPLAN = 292, /\* Get tampering detection arming schedule.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_TAMPERDETECTION\_WEEKPLAN = 293, /\* Set tampering detection arming schedule.

Set [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_INPUTSWITCHES\_WEEKPLAN = 294, /\* Get alarm input arming schedule.

See [NETDEV\_SWITCH\_WEEK\_PLAN\_INFO\_S](#_开关量计划（周）配置) \*/

NETDEV\_SET\_INPUTSWITCHES\_WEEKPLAN = 295, /\* Set alarm input arming schedule.

See [NETDEV\_SWITCH\_WEEK\_PLAN\_INFO\_S](#_开关量计划（周）配置) \*/

NETDEV\_GET\_OUTPUTSWITCHES\_WEEKPLAN = 296, /\* Get alarm output arming schedule.

See [NETDEV\_SWITCH\_WEEK\_PLAN\_INFO\_S](#_开关量计划（周）配置) \*/

NETDEV\_SET\_OUTPUTSWITCHES\_WEEKPLAN = 297, /\* Set alarm output arming schedule.

See [NETDEV\_SWITCH\_WEEK\_PLAN\_INFO\_S](#_开关量计划（周）配置) \*/

NETDEV\_GET\_DEFOCUSDETECTION\_WEEKPLAN = 298, /\* Get defocus detection arming schedule.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_DEFOCUSDETECTION\_WEEKPLAN = 299, /\* Set defocus detection arming schedule.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_SCENECHANGE\_WEEKPLAN = 300, /\* Get scene change arming schedule.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_SCENECHANGE\_WEEKPLAN = 301, /\* Set scene change arming schedule.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_FACEDETECTION\_WEEKPLAN = 302, /\* Get face detection arming schedule.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_FACEDETECTION\_WEEKPLAN = 303, /\* Set face detection arming schedule.

See [NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_RECORD\_STATUS = 320, /\* Get recording status info.

See [NETDEV\_RECORD\_STATUS\_LIST\_S](#_录像状态信息列表) \*/

NETDEV\_GET\_SERIAL\_CFG = 330, /\* Get serial interface configuration.

See [NETDEV\_SERIAL\_PORTS\_S](#_串口信息) \*/

NETDEV\_SET\_SERIAL\_CFG = 331, /\* Set serial interface configuration.

See [NETDEV\_SERIAL\_PORTS\_S](#_串口信息) \*/

NETDEV\_GET\_AUDIO\_DECODE\_STATUS = 340, /\* Get all audio decoding statuses.

See [NETDEV\_AUDIO\_DECODE\_STATUS\_LIST\_S](#_随路音频状态信息列表) \*/

NETDEV\_SET\_AUDIO\_DECODE\_STATUS = 341, /\* Set all audio decoding statuses.

See [NETDEV\_AUDIO\_DECODE\_STATUS\_LIST\_S](#_随路音频状态信息列表) \*/

NETDEV\_GET\_VIDEO\_LOSS\_RULE\_INFO = 400, /\* Get video loss detection configuration info.

See [NETDEV\_VIDEO\_LOSS\_RULE\_INFO\_S](#_视频丢失告警的配置信息) \*/

NETDEV\_SET\_VIDEO\_LOSS\_RULE\_INFO = 401, /\* Set video loss detection configuration info.

See [NETDEV\_VIDEO\_LOSS\_RULE\_INFO\_S](#_视频丢失告警的配置信息) \*/

NETDEV\_GET\_VIDEO\_LOSS\_LINKAGE\_ACTIONS = 410, /\* Get video loss alarm linkage actions.

See [NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表) \*/

NETDEV\_SET\_VIDEO\_LOSS\_LINKAGE\_ACTIONS = 411, /\* Set video loss alarm linkage actions.

See [NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表) \*/

NETDEV\_GET\_TAMPER\_DETECTION\_RULE\_INFO = 420, /\* Get tampering detection configuration info. See [NETDEV\_TAMPER\_DETECTION\_RULE\_INFO\_S](#_遮挡检测告警的配置信息) \*/

NETDEV\_SET\_TAMPER\_DETECTION\_RULE\_INFO = 421, /\* Set tampering detection configuration info. See [NETDEV\_TAMPER\_DETECTION\_RULE\_INFO\_S](#_遮挡检测告警的配置信息) \*/

NETDEV\_GET\_TAMPER\_DETECTION\_LINKAGE\_ACTIONS = 430, /\* Get tampering detection alarm linkage actions.

See [NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表) \*/

NETDEV\_SET\_TAMPER\_DETECTION\_LINKAGE\_ACTIONS = 431, /\* Set tampering detection alarm linkage actions. See [NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表) \*/

NETDEV\_GET\_TEMP\_DETECTION\_RULE\_INFO = 432, /\* Get temperature detection configuration info. See [NETDEV\_TEMP\_DETECTION\_RULE\_INFO\_S](#_温度检测告警的配置信息) \*/

NETDEV\_SET\_TEMP\_DETECTION\_RULE\_INFO = 433, /\* Set temperature detection configuration info. See [NETDEV\_TEMP\_DETECTION\_RULE\_INFO\_S](#_温度检测告警的配置信息) \*/

NETDEV\_GET\_TEMP\_DETECTION\_BLACKBODY\_INFO = 434, /\* Get temperature detection blackbody parameters. See [NETDEV\_TEMP\_DETECTION\_BLACKBODY\_INFO\_S](#_温度检测黑体信息结构体) \*/

NETDEV\_SET\_TEMP\_DETECTION\_BLACKBODY\_INFO = 435, /\* Set temperature detection blackbody parameters. See [NETDEV\_TEMP\_DETECTION\_BLACKBODY\_INFO\_S](#_温度检测黑体信息结构体) \*/

NETDEV\_GET\_TEMP\_DETECTION\_CORRECT\_INFO = 436, /\* Get temperature correction parameters. See [NETDEV\_TEMP\_DETECTION\_CORRECT\_INFO\_S](#_温度检测矫正信息结构体) \*/

NETDEV\_SET\_TEMP\_DETECTION\_CORRECT\_INFO = 437, /\* Set temperature correction parameters. See [NETDEV\_TEMP\_DETECTION\_CORRECT\_INFO\_S](#_温度检测矫正信息结构体) \*/

NETDEV\_GET\_TEMP\_DETECTION\_UNIT\_INFO = 438, /\* Get temperature detection unit. See [NETDEV\_TEMP\_DETECTION\_UNIT\_INFO\_S](#_温度检测单位信息结构体) \*/

NETDEV\_SET\_TEMP\_DETECTION\_UNIT\_INFO = 439, /\* Set temperature detection unit. See [NETDEV\_TEMP\_DETECTION\_UNIT\_INFO\_S](#_温度检测单位信息结构体) \*/

NETDEV\_GET\_MOTION\_DETECTION\_AREA\_TYPE = 440, /\* Get the area type of motion detection. See [NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_S](#_运动检测区域类型信息) \*/

NETDEV\_SET\_MOTION\_DETECTION\_AREA\_TYPE = 441, /\* Set the area type of motion detection. See [NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_S](#_运动检测区域类型信息) \*/

NETDEV\_GET\_MOTION\_DETECTION\_AREA\_GRID\_INFO = 442, /\* Get grid area info of motion detection. See [NETDEV\_MOTION\_DETECTION\_AREA\_GRID\_INFO\_S](#_运动检测宏块区域信息) \*/

NETDEV\_SET\_MOTION\_DETECTION\_AREA\_GRID\_INFO = 443, /\* Set grid area info of motion detection.

See [NETDEV\_MOTION\_DETECTION\_AREA\_GRID\_INFO\_S](#_运动检测宏块区域信息) \*/

NETDEV\_GET\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_LIST = 444, /\* Get all rectangle area info in motion detection.

See [NETDEV\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_LIST\_S](#_运动检测矩形区域信息列表) \*/

NETDEV\_SET\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_LIST = 445, /\* Set all rectangle area info in motion detection.

See [NETDEV\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_LIST\_S](#_运动检测矩形区域信息列表) \*/

NETDEV\_GET\_MOTION\_DETECTION\_LINKAGE\_ACTIONS = 450, /\* Get linkage actions of motion detection alarm. See [NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表)\*/

NETDEV\_SET\_MOTION\_DETECTION\_LINKAGE\_ACTIONS = 451, /\* Set linkage actions of motion detection alarm. See [NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表) \*/

NETDEV\_GET\_INPUT\_SWITCHES\_LINKAGE\_ACTIONS = 460, /\* Get linkage actions of alarm input (input switch alarm). See [NETDEV\_SWITCH\_LINKAGE\_ACTION\_S](#_开关量布控任务联动动作) \*/.

NETDEV\_SET\_INPUT\_SWITCHES\_LINKAGE\_ACTIONS = 461, /\* Set linkage actions of alarm input (input switch alarm). See [NETDEV\_SWITCH\_LINKAGE\_ACTION\_S](#_开关量布控任务联动动作) \*/

NETDEV\_GET\_RAID\_STATUS = 470, /\* Get RAID status. See [NETDEV\_RAID\_STATUS\_S](#_阵列状态信息) \*/

NETDEV\_GET\_RAID\_STORAGE\_CONTAINER\_INFO\_LIST = 471, /\* First use NETDEV\_GET\_RAID\_STATUS to get RAID status. If RAID is disabled, get storage container list.

See [NETDEV\_HDD\_INFO\_LIST\_S](#_磁盘信息列表) \*/

NETDEV\_GET\_STORAGE\_CONTAINER\_INFO\_LIST = 472, /\* First use NETDEV\_GET\_RAID\_STATUS to get RAID status. If RAID is disabled, get the storage container list.

See [NETDEV\_STORAGE\_CONTAINER\_INFO\_LIST\_S](#_存储容器信息列表) \*/

NETDEV\_GET\_HDD\_SMART\_INFO = 473, /\* Get S.M.A.R.T. info of specified HDD. See [NETDEV\_HDD\_SMART\_INFO\_S](#_硬盘Smart信息) \*/

NETDEV\_SET\_HDD\_SMART\_INFO = 474, /\* Set S.M.A.R.T. info of specified HDD. See [NETDEV\_HDD\_SMART\_CHECK\_INFO\_S](#_硬盘Smart检测信息) \*/

NETDEV\_GET\_RAID\_STATUS\_V30 = 475, /\* Get RAID status. Supported by VMS/NVR. See [NETDEV\_RAID\_STATUS\_S](#_阵列状态信息) \*/

NETDEV\_SET\_RAID\_STATUS\_V30 = 476, /\* Set RAID status. Supported by VMS/NVR. See [NETDEV\_RAID\_STATUS\_S](#_阵列状态信息) \*/

NETDEV\_GET\_CLOUD\_CONVENIENT\_ACCS\_STATUS = 480, /\* Get the status of quick adding mode of EZCloud devices. 0: Disable 1: Enable \*/

NETDEV\_SET\_CLOUD\_CONVENIENT\_ACCS\_STATUS = 481, /\* Set the status of quick adding mode of EZCloud devices. 0: Disable 1: Enable \*/

NETDEV\_GET\_PHOTO\_SERVER\_INFO = 482, /\* Get configuration info of photo server. See [NETDEV\_PHOTO\_SERVER\_CFG\_INFO\_LIST\_S](#_照片接收服务器配置信息列表) \*/

NETDEV\_SET\_PHOTO\_SERVER\_INFO = 483, /\* Set configuration info of photo server. See [NETDEV\_PHOTO\_SERVER\_CFG\_INFO\_LIST\_S](#_照片接收服务器配置信息列表) \*/

NETDEV\_GET\_ENHANCED\_CONFIG = 484, /\* Get enhanced configuration info of VMS. See [NETDEV\_ENHANCED\_CONFIG\_INFO\_S](#_功能增强配置信息) \*/

NETDEV\_SET\_ENHANCED\_CONFIG = 485, /\* Set enhanced configuration info of VMS. See [NETDEV\_ENHANCED\_CONFIG\_INFO\_S](#_功能增强配置信息) \*/

NETDEV\_GET\_INTELLIGENT\_SERVER\_INFO = 486, /\* Get intelligent server configuration. See [NETDEV\_DATA\_SERVER\_LIST](#_智能服务器配置列表) \*/

NETDEV\_SET\_INTELLIGENT\_SERVER\_INFO = 487, /\* Set intelligent server configuration. See [NETDEV\_DATA\_SERVER\_LIST](#_智能服务器配置列表) \*/

NETDEV\_GET\_MOTION\_INTERVAL\_INFO = 490, /\* Get alarm interval parameters of motion detection alarm. See [NETDEV\_MOTION\_INTERVAL\_INFO\_S](#_运动检测告警参数) \*/

NETDEV\_SET\_MOTION\_INTERVAL\_INFO = 491, /\* Set alarm interval parameters of motion detection alarm. See [NETDEV\_MOTION\_INTERVAL\_INFO\_S](#_运动检测告警参数) \*/

NETDEV\_GET\_MANAGER\_SERVER\_INFO = 492, /\* Get management server configuration. See [NETDEV\_MANAGER\_SERVER\_LIST\_S](#_管理服务器配置列表) \*/

NETDEV\_SET\_MANAGER\_SERVER\_INFO = 493, /\* Set management server configuration. See [NETDEV\_MANAGER\_SERVER\_LIST\_S](#_管理服务器配置列表) \*/

NETDEV\_GET\_VISIBLE\_RANGE\_INFO = 494, /\* Get visible range parameters. See [NETDEV\_VISIBLE\_RANGE\_INFO\_S](#_可视域参数信息) \*/

NETDEV\_SET\_VISIBLE\_RANGE\_INFO = 495, /\* Set visible range parameters. See [NETDEV\_VISIBLE\_RANGE\_INFO\_S](#_可视域参数信息) \*/

NETDEV\_GET\_RESET\_PWD\_VERTIFY\_INFO = 496, /\* Get verification info for retrieving user password. See [NETDEV\_RESET\_PWD\_VERTIFY\_INFO\_S](#_验证信息) \*/

NETDEV\_SET\_RESET\_PWD\_VERTIFY\_INFO = 497, /\* Set verification info for retrieving user password. See [NETDEV\_RESET\_PWD\_VERTIFY\_INFO\_S](#_验证信息)\*/

NETDEV\_GET\_CHL\_STORAGE\_QUOTA\_INFO = 498, /\* Get channel's storage quota info. See [NETDEV\_STORAGE\_QUOTA\_INFO\_S](#_存储配额信息) \*/

NETDEV\_SET\_CHL\_STORAGE\_QUOTA\_INFO = 499, /\* Set channel’s storage quota info. See [NETDEV\_STORAGE\_QUOTA\_INFO\_S](#_存储配额信息)  \*/

NETDEV\_GET\_CUSTOM\_PROTOCOL\_INFO = 501, /\* Get custom protocol configuration info. See [NETDEV\_CUSTON\_PROTOCOL\_INFO\_S](#_自定义协议信息) \*/

NETDEV\_SET\_CUSTOM\_PROTOCOL\_INFO = 502, /\* Set custom protocol configuration info. See [NETDEV\_CUSTON\_PROTOCOL\_INFO\_S](#_自定义协议信息) \*/

NETDEV\_GET\_CUSTOM\_PROTOCOL\_INFOLIST = 503, /\* Get custom protocol list. See [NETDEV\_CUSTON\_PROTOCOL\_SIMPLE\_INFO\_LIST\_S](#_自定义协议配置单例信息列表)\*/

NETDEV\_GET\_CHL\_STORAGE\_STRATEGY = 504, /\* Get channel’s storage strategy info. See [NETDEV\_STORAGE\_STRATEGY\_S](#_存储策略信息) \*/

NETDEV\_SET\_CHL\_STORAGE\_STRATEGY = 505, /\* Set channel’s storage strategy info. See [NETDEV\_STORAGE\_STRATEGY\_S](#_存储策略信息) \*/

NETDEV\_SET\_BACKFOCUS = 506, /\* Set backfocus. Applicable to IPC only. See  [NETDEV\_BACKFOCUS\_INFO\_S](#_后焦调节参数信息) \*/

NETDEV\_GET\_SMART\_ATTR\_MONITOR = 508, /\* Get smart attribute configuration. See [NETDEV\_SMART\_ATTR\_CFG\_S](#_智能属性配置) \*/

NETDEV\_SET\_SMART\_ATTR\_MONITOR = 509, /\* Set smart attribute configuration. See [NETDEV\_SMART\_ATTR\_CFG\_S](#_智能属性配置) \*/

NETDEV\_GET\_IMAGE\_CORRECT\_PARAM = 510, /\* Get image correction parameters. See [NETDEV\_IMAGE\_CORRECT\_PARAM\_S](#_图片矫正参数) \*/

NETDEV\_GET\_DEVICE\_TIME\_SYNCMODE = 512, /\* Get device’s time sync mode. See [NETDEV\_TIME\_SYNCMODE\_S](#_设备时间同步方式) \*/

NETDEV\_SET\_DEVICE\_TIME\_SYNCMODE = 513, /\* Set device’s time sync mode. See [NETDEV\_TIME\_SYNCMODE\_S](#_设备时间同步方式) \*/

NETDEV\_GET\_IMAGE\_SCENE\_AUTO\_SWITCH\_INFO = 514, /\* Get info of automatic scene switching. See [NETDEV\_SCENE\_AUTO\_SWITCH\_INFO\_S](#_场景自动切换开关信息) \*/

NETDEV\_SET\_IMAGE\_SCENE\_AUTO\_SWITCH\_INFO = 515, /\* Set info of automatic scene switching. [NETDEV\_SCENE\_AUTO\_SWITCH\_INFO\_S](#_场景自动切换开关信息) \*/

NETDEV\_GET\_IMAGE\_CURRENT\_SCENE\_TEMPLATE\_INFO = 516, /\* Get the current scene template in use. See [NETDEV\_SCENE\_TEMPLATE\_INFO\_S](#_场景模板信息) \*/

NETDEV\_SET\_IMAGE\_CURRENT\_SCENE\_TEMPLATE\_INFO = 517, /\* Set the current scene template in use. See [NETDEV\_SCENE\_TEMPLATE\_INFO\_S](#_场景模板信息) \*/

NETDEV\_GET\_IMAGE\_DEFAULT\_SCENE\_TEMPLATE\_INFO = 518, /\* Get the default scene template. See [NETDEV\_SCENE\_TEMPLATE\_INFO\_S](#_场景模板信息) \*/

NETDEV\_SET\_IMAGE\_DEFAULT\_SCENE\_TEMPLATE\_INFO = 519, /\* Set the default scene template. See [NETDEV\_SCENE\_TEMPLATE\_INFO\_S](#_场景模板信息) \*/

NETDEV\_GET\_IMAGE\_ALL\_SCENE\_INFO = 520, /\* Get all scene info. See [NETDEV\_SCENE\_INFO\_LIST\_S](#_所有场景信息) \*/

NETDEV\_SET\_IMAGE\_ALL\_SCENE\_INFO = 521, /\* Set all scene info. See [NETDEV\_SCENE\_INFO\_LIST\_S](#_所有场景信息) \*/

NETDEV\_GET\_IMAGE\_SPECIFIC\_SCENE\_INFO = 522, /\* Get info of a specified scene template. See [NETDEV\_SCENE\_INFO\_S](#_场景信息) \*/

NETDEV\_SET\_IMAGE\_SPECIFIC\_SCENE\_INFO = 523, /\* Set info of a specified scene template. See [NETDEV\_SCENE\_INFO\_S](#_场景信息) \*/

NETDEV\_GET\_ATTR\_COLLECT\_INFO = 524, /\* Get attribute collection info. See [NETDEV\_ATTR\_COLLECT\_INFO\_S](#_属性采集信息) \*/

NETDEV\_SET\_ATTR\_COLLECT\_INFO = 525, /\* Set attribute collection info. See [NETDEV\_ATTR\_COLLECT\_INFO\_S](#_属性采集信息) \*/

NETDEV\_GET\_SYS\_TIMECFG = 526, /\* Get system time parameters. See [NETDEV\_TIME\_CFG\_V30\_S](#_系统时间配置). Currently only display control devices support URL. When channel ID is not required, xFF must be passed in as channel ID. \*/

NETDEV\_SET\_SYS\_TIMECFG = 527, /\* Set system time parameters. See [NETDEV\_TIME\_CFG\_V30\_S](#_系统时间配置). Currently only display control devices support URL. When channel ID is not needed, NETDEV\_INVALID\_CHANNEL\_ID must be passed in as the channel ID. \*/

NETDEV\_GET\_NTPCFG\_V30 = 528, /\* Get all NTP server info. See [NETDEV\_SYSTEM\_NTP\_INFO\_LIST\_S](#_NTP列表). Currently only display control devices support URL. When channel ID is not needed, NETDEV\_INVALID\_CHANNEL\_ID must be passed in as the channel ID. \*/

NETDEV\_SET\_NTPCFG\_V30 = 529, /\* Set all NTP server info. See [NETDEV\_SYSTEM\_NTP\_INFO\_LIST\_S](#_NTP列表). Current only display control device support URL. When channel ID is not needed, NETDEV\_INVALID\_CHANNEL\_ID must be passed in as the channel ID \*/

NETDEV\_GET\_DEVICE\_TIME\_SYNCMODE\_EX = 530, /\* Get device’s time sync mode. See [NETDEV\_TIME\_SYNCMODE\_S](#_设备时间同步方式). Currently only display control devices support URL. When channel ID is not required, NETDEV\_INVALID\_CHANNEL\_ID must be passed in as the channel ID. \*/

NETDEV\_SET\_DEVICE\_TIME\_SYNCMODE\_EX = 531, /\* Set device’s time sync mode. See [NETDEV\_TIME\_SYNCMODE\_S](#_设备时间同步方式). Currently only display control devices support URL. When channel ID is not required, NETDEV\_INVALID\_CHANNEL\_ID must be passed in as the channel ID. \*/

NETDEV\_GET\_AREA\_PEOPLE\_COUNT\_RULE\_INFO = 532, /\* Get configuration info of area people counting rules. See [NETDEV\_AREA\_PEOPLE\_COUNT\_RULE\_INFO\_S](#_区域人数统计规则信息) \*/

NETDEV\_SET\_AREA\_PEOPLE\_COUNT\_RULE\_INFO = 533, /\* Set configuration info of area people counting rules. See [NETDEV\_AREA\_PEOPLE\_COUNT\_RULE\_INFO\_S](#_区域人数统计规则信息) \*/

NETDEV\_GET\_LINE\_PEOPLE\_COUNT\_RULE\_INFO = 534, /\* Get configuration info of tripwire people counting rules. See [NETDEV\_LINE\_PEOPLE\_COUNT\_RULE\_INFO\_S](#_绊线人数统计规则信息) \*/

NETDEV\_SET\_LINE\_PEOPLE\_COUNT\_RULE\_INFO = 535, /\* Set configuration info of tripwire people counting rules. See [NETDEV\_LINE\_PEOPLE\_COUNT\_RULE\_INFO\_S](#_绊线人数统计规则信息) \*/

NETDEV\_GET\_LINE\_PEOPLE\_COUNT\_RESET\_INFO = 536, /\* Get reset configuration of tripwire people counting. See [NETDEV\_LINE\_PEOPLE\_COUNT\_RESET\_INFO\_S](#_绊线人数统计清零信息) \*/

NETDEV\_SET\_LINE\_PEOPLE\_COUNT\_RESET\_INFO = 537, /\* Set reset configuration of tripwire people counting. See [NETDEV\_LINE\_PEOPLE\_COUNT\_RESET\_INFO\_S](#_绊线人数统计清零信息) \*/

NETDEV\_GET\_BOX\_DOME\_LINKAGE\_INFO = 547, /\* Get info about dome camera that has been added on box camera. See [NETDEV\_LINK\_DOME\_INFO\_LIST\_S](#_联动球机信息列表) \*/

NETDEV\_CFG\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_CONFIG\_COMMAND\_E;

### IP protocol types

typedef enum tagHOSTType

{

NETDEV\_NETWORK\_HOSTTYPE\_IPV4 = 0, /\* IPv4 \*/

NETDEV\_NETWORK\_HOSTTYPE\_IPV6 = 1, /\* IPv6 \*/

NETDEV\_NETWORK\_HOSTTYPE\_DNS = 2, /\* Domain name\*/

NETDEV\_NETWORK\_HOSTTYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HOSTTYPE\_E;

### Image quality

typedef enum tagNETDEVVideoQuality

{

NETDEV\_VQ\_L0 = 0, /\* Highest \*/

NETDEV\_VQ\_L1 = 1,

NETDEV\_VQ\_L2 = 4, /\* Higher \*/

NETDEV\_VQ\_L3 = 8,

NETDEV\_VQ\_L4 = 12, /\* Medium \*/

NETDEV\_VQ\_L5 = 16,

NETDEV\_VQ\_L6 = 20, /\* Low \*/

NETDEV\_VQ\_L7 = 24,

NETDEV\_VQ\_L8 = 28, /\* Very low \*/

NETDEV\_VQ\_L9 = 31, /\* Lowest \*/

NETDEV\_VQ\_LEVEL\_INVALID = -1 /\* Valid value \*/

}NETDEV\_VIDEO\_QUALITY\_E;

### Main stream types

typedef enum tagNETDEVMainStreamType

{

NETDEV\_MAIN\_STREAM\_TYPE\_TIME = 0, /\* Schedule \*/

NETDEV\_MAIN\_STREAM\_TYPE\_EVENT = 1 /\* Event \*/

}NETDEV\_MAIN\_STREAM\_TYPE\_E;

### Bitrate types

typedef enum tagNETDEVBitRateType

{

NETDEV\_BIT\_RATE\_TYPE\_CBR = 0, /\* CBR \*/

NETDEV\_BIT\_RATE\_TYPE\_VBR = 1 /\* VBR \*/

}NETDEV\_BIT\_RATE\_TYPE\_E;

### GOP types

typedef enum tagNETDEVGopType

{

NETDEV\_GOP\_TYPE\_IP = 0, /\* IP \*/

NETDEV\_GOP\_TYPE\_IBP = 1, /\* IBP \*/

NETDEV\_GOP\_TYPE\_IBBP = 2, /\* IBBP \*/

NETDEV\_GOP\_TYPE\_I = 3 /\* I \*/

}NETDEV\_GOP\_TYPE\_E;

### Extended image encoding modes

typedef enum tagNETDEVSmartEncodeMode

{

NETDEV\_SMART\_ENCODE\_MODE\_OFF = 0, /\* Off \*/

NETDEV\_SMART\_ENCODE\_MODE\_BASIC = 1, /\* Basic smart encoding \*/

NETDEV\_SMART\_ENCODE\_MODE\_ADVANCED = 2 /\* Advanced smart encoding \*/

}NETDEV\_SMART\_ENCODE\_MODE\_E;

### Enumeration of OSD time formats

typedef enum tagNETDEVTimeOSDFormateCap

{

NETDEV\_OSD\_TIME\_FORMAT\_CAP\_HHMMSS = 0, /\* HH:mm:ss \*/

NETDEV\_OSD\_TIME\_FORMAT\_CAP\_HH\_MM\_SS\_PM = 1 /\* hh:mm:ss tt \*/

}NETDEV\_OSD\_TIME\_FORMAT\_CAP\_E;

### OSD types

typedef enum tagNETDEVOSDTextType

{

NETDEV\_OSD\_TEXT\_DATAANDTIME = 0, /\* Date and time, for example, yyyy-MM-dd.HH:mm:ss. For date format, see [NETDEV\_OSD\_DATE\_FORMAT\_CAP\_E](#_OSD日期格式能力集). For time format, see [NETDEV\_OSD\_TIME\_FORMAT\_CAP\_E](#_OSD时间格式能力集枚举) \*/

NETDEV\_OSD\_TEXT\_PLAIN

}NETDEV\_OSD\_TEXT\_TYPE\_E;

### Enumeration of OSD font styles

typedef enum tagNETDEVOSDFontStyle

{

NETDEV\_OSD\_FONT\_STYLE\_BACKGROUND = 0, /\* Background \*/

NETDEV\_OSD\_FONT\_STYLE\_STROKE = 1, /\* Stroke \*/

NETDEV\_OSD\_FONT\_STYLE\_HOLLOW = 2, /\* Hollow \*/

NETDEV\_OSD\_FONT\_STYLE\_NORMAL = 3 /\* Normal \*/

}NETDEV\_OSD\_FONT\_STYLE\_E;

### Enumeration of OSD font sizes

typedef enum tagNETDEVOSDFontSize

{

NETDEV\_OSD\_FONT\_SIZE\_LARGE = 0, /\* Very large \*/

NETDEV\_OSD\_FONT\_SIZE\_BIG = 1, /\* Large \*/

NETDEV\_OSD\_FONT\_SIZE\_MEDIUM = 2, /\* Medium \*/

NETDEV\_OSD\_FONT\_SIZE\_SMALL = 3 /\* Small \*/

}NETDEV\_OSD\_FONT\_SIZE\_E;

### Date format

typedef enum tagNETDEVDateOSDFormate

{

NETDEV\_OSD\_DATE\_FORMAT\_YYYY\_MMDD = 0, /\* yyyy-MM-dd \*/

NETDEV\_OSD\_DATE\_FORMAT\_MMDD\_YYYY = 1, /\* MM-dd-yyyy \*/

NETDEV\_OSD\_DATE\_FORMAT\_CHINESE\_YYYY\_MMDD = 2, /\* yyyyMMddyyyy/MM/dd\*/

NETDEV\_OSD\_DATE\_FORMAT\_CHINESE\_MMDD\_YYYY = 3, /\* MMddyyyyMM/dd/yyyy\*/

NETDEV\_OSD\_DATE\_FORMAT\_CHINESE\_YYY\_MMDD\_X = 4, /\* yyyyMMddDay (Day: e.g., Monday) dddd yyyy MM dd\*/

NETDEV\_OSD\_DATE\_FORMAT\_CHINESE\_MMDD\_YYYY\_X = 5, /\* MMddyyyyDay (Day: e.g., Monday) dddd MM dd yyyy\*/

NETDEV\_OSD\_DATE\_FORMAT\_ENGLISH\_DDMM\_YYYY = 100, /\* dd/MM/yyyy \*/

NETDEV\_OSD\_DATE\_FORMAT\_ENGLISH\_MMDD\_YYYY = 101, /\* MM/dd/yyyy \*/

NETDEV\_OSD\_DATE\_FORMAT\_ENGLISH\_DDMMMM\_YYYY = 102, /\* dd MMMM, yyyy \*/

NETDEV\_OSD\_DATE\_FORMAT\_ENGLISH\_MMMMDD\_YYYY = 103, /\* MMMM dd, yyyy \*/

NETDEV\_OSD\_DATE\_FORMAT\_ENGLISH\_DDDDDDMMMM\_YYYY = 104, /\* dddd, dd MMMM, yyyy \*/

NETDEV\_OSD\_DATE\_FORMAT\_ENGLISH\_DDDDDDMMMM\_DDYYYY = 105, /\* dddd, MMMM dd, yyyy \*/

NETDEV\_OSD\_DATE\_FORMAT\_ENGLISH\_YYYY\_MMDD = 106, /\* yyyy/MM/dd \*/

NETDEV\_OSD\_DATE\_FORMAT\_ENGLISH\_YYYY\_MMMMDD = 107, /\* yyyy, MMMM dd \*/

NETDEV\_OSD\_DATE\_FORMAT\_ENGLISH\_DDDDYY\_MMDD = 108 /\* dddd, yy, MM dd \*/

}NETDEV\_OSD\_DATE\_FORMAT\_E;

### Enumeration of OSD time formats

typedef enum tagNETDEVTimeOSDFormate

{

NETDEV\_OSD\_TIME\_FORMAT\_HHMMSS = 0, /\* HH:mm:ss \*/

NETDEV\_OSD\_TIME\_FORMAT\_HH\_MM\_SS\_T = 1, /\* hh:mm:ss t.t. \*/

NETDEV\_OSD\_TIME\_FORMAT\_HH\_MM\_SS\_TT = 2, /\* hh:mm:ss tt \*/

NETDEV\_OSD\_TIME\_FORMAT\_TT\_HH\_MM\_SS = 3, /\* tt hh:mm:ss \*/

NETDEV\_OSD\_TIME\_FORMAT\_PM\_HH\_MM\_SS = 4, /\* 下午hh:mm:ss P.M. hh:mm:ss\*/

NETDEV\_OSD\_TIME\_FORMAT\_HH\_MM\_SS\_XX = 5, /\* HH:mm:ss.xxx \*/

NETDEV\_OSD\_TIME\_FORMAT\_HH\_MM\_SS\_XX\_TT = 6 /\* hh:mm:ss.xxx tt \*/

}NETDEV\_OSD\_TIME\_FORMAT\_E;

### Enumeration of OSD alignments

typedef enum tagNETDEVOSDAlign

{

NETDEV\_OSD\_ALIGN\_LEFT = 0, /\* Left alignment \*/

NETDEV\_OSD\_ALIGN\_RIGHT = 1 /\* Right alignment \*/

}NETDEV\_OSD\_ALIGN\_E;

### Enumeration of OSD margin sizes

typedef enum tagNETDEVOSDMinMargin

{

NETDEV\_OSD\_MIN\_MARGIN\_NONE = 0, /\* None \*/

NETDEV\_OSD\_MIN\_MARGIN\_SINGLE = 1, /\* 1-character width \*/

NETDEV\_OSD\_MIN\_MARGIN\_DOUBLE = 2 /\* 2-character width \*/

}NETDEV\_OSD\_MIN\_MARGIN\_E;

### Operation modes of alarm input

typedef enum tagNETDEVBooleanMode

{

NETDEV\_BOOLEAN\_MODE\_OPEN = 1, /\* Normally open \*/

NETDEV\_BOOLEAN\_MODE\_CLOSE = 2, /\* Normally closed \*/

NETDEV\_BOOLEAN\_MODE\_INVALID

}NETDEV\_BOOLEAN\_MODE\_E;

### Commands used to control manual alarm output

typedef enum tagNETDEVOutPutManualAlarmCmd

{

NETDEV\_MANUAL\_ALARM\_CMD\_CLOSE = 0, /\* Off \*/

NETDEV\_MANUAL\_ALARM\_CMD\_TRIGGER = 1 /\* Trigger \*/

}NETDEV\_MANUAL\_ALARM\_CMD\_E;

### Exposure modes

typedef enum tagNETDEVExposureMode

{

NETDEV\_EXPOSURE\_MODE\_AUTOMATIC = 0, /\* Automatic exposure \*/

NETDEV\_EXPOSURE\_MODE\_CUSTOM = 1, /\* Custom exposure \*/

NETDEV\_EXPOSURE\_MODE\_SHUTTER\_PRIORITY = 2, /\* Shutter priority \*/

NETDEV\_EXPOSURE\_MODE\_APERTURE\_PRIORITY = 3, /\* Aperture priority \*/

NETDEV\_EXPOSURE\_MODE\_GAIN\_PRIORITY = 4, /\* Gain priority \*/

NETDEV\_EXPOSURE\_MODE\_INDOOR\_50\_HZ = 5, /\* Indoor 50Hz \*/

NETDEV\_EXPOSURE\_MODE\_INDOOR\_60\_HZ = 6, /\* Indoor 60Hz \*/

NETDEV\_EXPOSURE\_MODE\_MANUAL\_OPERATION = 7, /\* Manual exposure \*/

NETDEV\_EXPOSURE\_MODE\_LOWER\_GHOSTING = 8, /\* Low motion blur \*/

NETDEV\_EXPOSURE\_MODE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_EXPOSURE\_MODE\_E;

### Supported shutter times

typedef enum tagNETDEVShutterTimeRange

{

NETDEV\_SHUTTER\_TIME\_AUTOMATIC = 0, /\* Shutter time is “auto” \*/

NETDEV\_SHUTTER\_TIME\_1 = 1, /\* 1/1 \*/

NETDEV\_SHUTTER\_TIME\_1\_2 = 2, /\* 1/2 \*/

NETDEV\_SHUTTER\_TIME\_1\_3 = 3, /\* 1/3 \*/

NETDEV\_SHUTTER\_TIME\_1\_4 = 4, /\* 1/4 \*/

NETDEV\_SHUTTER\_TIME\_1\_6 = 6, /\* 1/6 \*/

NETDEV\_SHUTTER\_TIME\_1\_8 = 8, /\* 1/8 \*/

NETDEV\_SHUTTER\_TIME\_1\_10 = 10, /\* 1/10\*/

NETDEV\_SHUTTER\_TIME\_1\_12 = 12, /\* 1/12\*/

NETDEV\_SHUTTER\_TIME\_1\_15 = 15, /\* 1/15\*/

NETDEV\_SHUTTER\_TIME\_1\_20 = 20, /\* 1/20\*/

NETDEV\_SHUTTER\_TIME\_1\_25 = 25, /\* 1/25\*/

NETDEV\_SHUTTER\_TIME\_1\_30 = 30, /\* 1/30\*/

NETDEV\_SHUTTER\_TIME\_1\_50 = 50, /\* 1/50\*/

NETDEV\_SHUTTER\_TIME\_1\_60 = 60, /\* 1/60\*/

NETDEV\_SHUTTER\_TIME\_1\_100 = 100, /\* 1/100\*/

NETDEV\_SHUTTER\_TIME\_1\_120 = 120, /\* 1/120\*/

NETDEV\_SHUTTER\_TIME\_1\_150 = 150, /\* 1/150\*/

NETDEV\_SHUTTER\_TIME\_1\_180 = 180, /\* 1/180\*/

NETDEV\_SHUTTER\_TIME\_1\_200 = 200, /\* 1/200\*/

NETDEV\_SHUTTER\_TIME\_1\_250 = 250, /\* 1/250\*/

NETDEV\_SHUTTER\_TIME\_1\_500 = 500, /\* 1/500\*/

NETDEV\_SHUTTER\_TIME\_1\_1000 = 1000, /\* 1/1000\*/

NETDEV\_SHUTTER\_TIME\_1\_2000 = 2000, /\* 1/2000\*/

NETDEV\_SHUTTER\_TIME\_1\_4000 = 4000, /\* 1/4000\*/

NETDEV\_SHUTTER\_TIME\_1\_8000 = 8000, /\* 1/8000\*/

NETDEV\_SHUTTER\_TIME\_1\_50000 = 50000, /\* 1/50000\*/

NETDEV\_SHUTTER\_TIME\_1\_100000 = 100000, /\* 1/100000\*/

NETDEV\_SHUTTER\_TIME\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_SHUTTER\_TIME\_RANGE\_E;

### Metering control modes

typedef enum tagNETDEVWideDynamicMode

{

NETDEV\_WDM\_CLOSED = 0, /\* Off \*/

NETDEV\_WDM\_OPEN = 1, /\* On \*/

NETDEV\_WDM\_AUTO = 2, /\* Auto \*/

NETDEV\_WDM\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_WIDE\_DYNAMIC\_MODE\_E;

### Day/night mode types

typedef enum tagNETDEVDayNightMode

{

NETDEV\_DNM\_AUTO = 0, /\* Auto mode \*/

NETDEV\_DNM\_DAY\_COLOR = 1, /\* Day--color \*/

NETDEV\_DNM\_NIGHT\_BAW = 2, /\* Night--black & white \*/

NETDEV\_DNM\_LINK\_COLOR\_TO\_BLACK = 3, /\* Trigger switching from color to black & white \*/

NETDEV\_DNM\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_DAY\_NIGHT\_MODE\_E;

### Illuminator types

typedef enum tagNETDEVLampType

{

NETDEV\_LAMP\_TYPE\_WITHOUT = 0, /\* No illuminator \*/

NETDEV\_LAMP\_TYPE\_WHITE = 1, /\* White light \*/

NETDEV\_LAMP\_TYPE\_INFRARED = 2, /\* IR light \*/

NETDEV\_LAMP\_TYPE\_LASER\_DEVICE = 3, /\* Laser \*/

NETDEV\_LAMP\_TYPE\_INFRARED\_SHIELD = 4, /\* IR light with housing \*/

NETDEV\_LAMP\_TYPE\_EXTERNAL\_HEATING\_LIGHT = 5, /\* External warm light \*/

NETDEV\_LAMP\_TYPE\_DOUBLE\_LIGHT = 6, /\* Dual light sources (white light + IR) \*/

NETDEV\_LAMP\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_LAMP\_TYPE\_E;

### White balance modes

typedef enum tagNETDEVWhiteBalance

{

NETDEV\_WBM\_AUTO = 0, /\*Auto \*/

NETDEV\_WBM\_TRIMMING = 1, /\* Fine-tune \*/

NETDEV\_WBM\_INDOOR = 2, /\* Indoor \*/

NETDEV\_WBM\_OUTDOOR = 3, /\* Outdoor \*/

NETDEV\_WBM\_AUTO\_MERCURY\_LAMP = 4, /\* Auto mercury lamp \*/

NETDEV\_WBM\_NIGHT\_TRIMMING = 5, /\* Fine-tune (based on night mode) \*/

NETDEV\_WBM\_AUTO\_NA\_LAMP = 6, /\* Auto sodium lamp \*/

NETDEV\_WBM\_LOCK = 7, /\* Lock \*/

NETDEV\_WBM\_SELF\_ADAPT\_NIGHT\_TRIMMING = 8, /\* Adaptive fine-tune (based on night mode) \*/

NETDEV\_WBM\_AUTO\_SECOND = 9, /\* Auto 2 \*/

NETDEV\_WBM\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_WHITE\_BALANCE\_MODE\_E;

### Enumeration of HDD working modes

typedef enum tagNETDEVDiskWorkStatus

{

NETDEV\_DISK\_WORK\_STATUS\_EMPTY = 0, /\* Empty \*/

NETDEV\_DISK\_WORK\_STATUS\_UNFORMAT = 1, /\* Unformatted \*/

NETDEV\_DISK\_WORK\_STATUS\_FORMATING = 2, /\* Formatting \*/

NETDEV\_DISK\_WORK\_STATUS\_RUNNING = 3, /\* Running \*/

NETDEV\_DISK\_WORK\_STATUS\_HIBERNATE = 4, /\* Hibernate \*/

NETDEV\_DISK\_WORK\_STATUS\_ABNORMAL = 5, /\* Abnormal \*/

NETDEV\_DISK\_WORK\_STATUS\_UNKNOWN = 6, /\* Unknown \*/

NETDEV\_DISK\_WORK\_STATUS\_INVALID /\* Invalid value \*/

}NETDEV\_DISK\_WORK\_STATUS\_E;

### Focus modes

typedef enum tagNETDEVFocusMode

{

NETDEV\_FOCUS\_AUTO = 1, /\* Auto focus \*/

NETDEV\_FOCUS\_MANUAL = 2 /\* Manual focus \*/

}NETDEV\_FOCUS\_MODE\_E;

### Day/night mode

typedef enum tagNETDEVIrCutFilterMode

{

NETDEV\_IR\_CUT\_FILTER\_ON = 0, /\* Day mode \*/

NETDEV\_IR\_CUT\_FILTER\_OFF = 1, /\* Night mode \*/

NETDEV\_IR\_CUT\_FILTER\_AUTO = 2 /\* Auto mode \*/

}NETDEV\_IR\_CUT\_FILTER\_MODE\_E;

### Enumeration of defog modes

typedef enum tagNETDEVDefoggingMode

{

NETDEV\_DEFOGGING\_ON = 0, /\* On \*/

NETDEV\_DEFOGGING\_OFF=1 /\* Off \*/

}NETDEV\_DEFOGGING\_MODE\_E;

### Enumeration of audio collection ports

typedef enum tagNETDEVAudioCollectionPortType

{

NETDEV\_AUDIO\_COLLECTION\_PORT\_TYPE\_AUDIO\_IN = 0, /\* Audio in \*/

NETDEV\_AUDIO\_COLLECTION\_PORT\_TYPE\_SERIAL\_IN = 1, /\* Serial port \*/

NETDEV\_AUDIO\_COLLECTION\_PORT\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_AUDIO\_COLLECTION\_PORT\_TYPE\_E;

### Enumeration of audio encoding formats

typedef enum tagNETDEVAudioEncodeFormatType

{

NETDEV\_AUDIO\_ENCODE\_FORMAT\_TYPE\_MPEG1 = 0, /\* MPEG1 \*/

NETDEV\_AUDIO\_ENCODE\_FORMAT\_TYPE\_G711A = 1, /\* G.711A \*/

NETDEV\_AUDIO\_ENCODE\_FORMAT\_TYPE\_G711U = 2, /\* G.711U \*/

NETDEV\_AUDIO\_ENCODE\_FORMAT\_TYPE\_ADPCM = 3, /\* ADPCM \*/

NETDEV\_AUDIO\_ENCODE\_FORMAT\_TYPE\_OGG = 4, /\* OGG \*/

NETDEV\_AUDIO\_ENCODE\_FORMAT\_TYPE\_Auto = 5, /\* Auto \*/

NETDEV\_AUDIO\_ENCODE\_FORMAT\_TYPE\_AACLC = 6, /\* AAC-LC \*/

NETDEV\_AUDIO\_ENCODE\_FORMAT\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_AUDIO\_ENCODE\_FORMAT\_TYPE\_E;

### Enumeration of audio sampling rates

typedef enum tagNETDEVAudioSamplingRate

{

NETDEV\_AUDIO\_SAMPLING\_RATE\_8KHZ = 0, /\* 8KHz \*/

NETDEV\_AUDIO\_SAMPLING\_RATE\_16KHZ = 1, /\* 16KHz \*/

NETDEV\_AUDIO\_SAMPLING\_RATE\_22\_05KHZ = 2, /\* 22.05KHz \*/

NETDEV\_AUDIO\_SAMPLING\_RATE\_24KHZ = 3, /\* 24KHz \*/

NETDEV\_AUDIO\_SAMPLING\_RATE\_32KHZ = 4, /\* 32KHz \*/

NETDEV\_AUDIO\_SAMPLING\_RATE\_44\_1KHZ = 5, /\* 44.1KHz \*/

NETDEV\_AUDIO\_SAMPLING\_RATE\_48KHZ = 6, /\* 48KHz \*/

NETDEV\_AUDIO\_SAMPLING\_RATE\_96KHZ = 7, /\* 96KHz \*/

NETDEV\_AUDIO\_SAMPLING\_RATE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_AUDIO\_SAMPLING\_RATE\_E;

### Enumeration of audio input modes

typedef enum tagNETDEVAudioInChlMode

{

NETDEV\_AUDIO\_CHL\_MODE\_LINE = 0, /\* Line \*/

NETDEV\_AUDIO\_CHL\_MODE\_MIC = 1, /\* MIC \*/

NETDEV\_AUDIO\_CHL\_MODE\_RS485 = 2, /\* RS485 sound pickup \*/

NETDEV\_AUDIO\_CHL\_MODE\_SPEAKER = 3, /\* Speaker \*/

NETDEV\_AUDIO\_CHL\_MODE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_AUDIO\_CHL\_MODE\_E;

### DST offset time

typedef enum tagNetDEVDSTOffsetTime

{

NETDEV\_DST\_OFFSET\_TIME\_30MIN = 30,

NETDEV\_DST\_OFFSET\_TIME\_60MIN = 60,

NETDEV\_DST\_OFFSET\_TIME\_90MIN = 90,

NETDEV\_DST\_OFFSET\_TIME\_120MIN = 120,

NETDEV\_DST\_OFFSET\_TIME\_INVALID = 0xff

}NETDEV\_DST\_OFFSET\_TIME;

### Week

typedef enum tagNetDEVDayInWeek

{

NETDEV\_WEEK\_SUNDAY = 0, /\* Sunday \*/

NETDEV\_WEEK\_MONDAY = 1, /\* Monday \*/

NETDEV\_WEEK\_TUESDAY = 2, /\* Tuesday \*/

NETDEV\_WEEK\_WEDNESDAY = 3, /\* Wednesday \*/

NETDEV\_WEEK\_THURSDAY = 4, /\* Thursday \*/

NETDEV\_WEEK\_FRIDAY = 5, /\* Friday \*/

NETDEV\_WEEK\_SATURDAY = 6, /\* Saturday \*/

NETDEV\_WEEK\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_DAY\_IN\_WEEK\_E;

### Enumeration pre-alarm recording time

typedef enum tagNETDEVPreRecordTime

{

NETDEV\_PRE\_RECORD\_TIME\_ZERO = 0, /\* 0s \*/

NETDEV\_PRE\_RECORD\_TIME\_FIVE = 5, /\* 5s \*/

NETDEV\_PRE\_RECORD\_TIME\_TEN = 10, /\* 10s \*/

NETDEV\_PRE\_RECORD\_TIME\_TWENTY = 20, /\* 20s \*/

NETDEV\_PRE\_RECORD\_TIME\_THIRTY = 30, /\* 30s \*/

NETDEV\_PRE\_RECORD\_TIME\_SIXTY = 60 /\* 60s \*/

}NETDEV\_PRE\_RECORD\_TIME\_E;

### Enumeration of post-alarm recording time

typedef enum tagNETDEVPostRecordTime

{

NETDEV\_POST\_RECORD\_TIME\_FIVE =5, /\* 5s\*/

NETDEV\_POST\_RECORD\_TIME\_TEN =10, /\* 10s \*/

NETDEV\_POST\_RECORD\_TIME\_THIRTY =30, /\* 30s \*/

NETDEV\_POST\_RECORD\_TIME\_SIXTY =60, /\* 60s \*/

NETDEV\_POST\_RECORD\_TIME\_ONE\_HUNDRED\_AND\_TWENTY =120, /\* 120s \*/

NETDEV\_POST\_RECORD\_TIME\_THREE\_HUNDRED =300, /\* 300s \*/

NETDEV\_POST\_RECORD\_TIME\_SIX\_HUNDRED =600 /\* 600s \*/

}NETDEV\_POST\_RECORD\_TIME\_E;

### Enumeration of motion detection area types

typedef enum tagNETDEVMotionDetectAreaType

{

NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_RECTANGLE = 0, /\* Rectangle area \*/

NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_MB = 1, /\* Macroblock \*/

NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_E;

### HDD types

typedef enum tagNETDEVHDDType

{

NETDEV\_HDD\_TYPE\_LOCAL\_HDD = 0, /\* LocalHDD \*/

NETDEV\_HDD\_TYPE\_NO1\_EXTEND\_HDD = 1, /\* No1ExtendCabinetHDD \*/

NETDEV\_HDD\_TYPE\_NO2\_EXTEND\_HDD = 2, /\* No2ExtendCabinetHDD \*/

NETDEV\_HDD\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HDD\_TYPE\_E;

### HDD working modes

typedef enum tagNETDEVHDDWorkMode

{

NETDEV\_HDD\_WORK\_MODE\_COMMON = 0, /\* Common disk \*/

NETDEV\_HDD\_WORK\_MODE\_RAID = 1, /\* RAID disk \*/

NETDEV\_HDD\_WORK\_MODE\_HOT\_BACKUP = 2, /\* Hot spare disk \*/

NETDEV\_HDD\_WORK\_MODE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HDD\_WORK\_MODE\_E;

### HDD status

typedef enum tagNETDEVHDDStatus

{

NETDEV\_HDD\_STATUS\_NO = 0, /\* No HDD \*/

NETDEV\_HDD\_STATUS\_NORMAL = 1, /\* Normal \*/

NETDEV\_HDD\_STATUS\_ABNORMAL = 2, /\* Abnormal \*/

NETDEV\_HDD\_STATUS\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HDD\_STATUS\_E;

### Storage container status

typedef enum tagNETDEVStorageContainerStatus

{

NETDEV\_STORAGE\_CONTAINER\_STATUS\_NO = 0, /\* No HDD/Idle \*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_UNFORMATTED = 1, /\* Unformatted \*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_FORMATTING = 2, /\* Formatting \*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_NORMAL = 3, /\* Normal \*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_SLEEP = 4, /\* Sleep \*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_ABNORMAL = 5, /\* Abnormal \*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_SWITCH = 6, /\* Switching \*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_UNINSTALLED = 7, /\* Unmounted \*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_STORAGE\_CONTAINER\_STATUS\_E;

### Storage container attributes

typedef enum tagNETDEVStorageContainerProperty

{

NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_RW = 0, /\* Read/Write \*/

NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_R = 1, /\* Read-only \*/

NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_REDUNDANT = 2, /\* Redundant \*/

NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_E;

### Address types

typedef enum tagNETDEVAddrType

{

NETDEV\_ADDR\_TYPE\_IPV4 = 0, /\* IPv4 \*/

NETDEV\_ADDR\_TYPE\_IPV6 = 1, /\* IPv6 \*/

NETDEV\_ADDR\_TYPE\_DOMAIN = 2, /\* Domain name \*/

NETDEV\_ADDR\_TYPE\_IPV4\_IPV6 = 3, /\* IPv4 and IPv6 \*/

NETDEV\_ADDR\_TYPE\_INVALID = 0XFFFF /\* Invalid value \*/

}NETDEV\_ADDR\_TYPE\_E;

### Storage container usage

typedef enum tagNETDEVStorageContainerUsageType

{

NETDEV\_STORAGE\_CONTAINER\_USAGE\_TYPE\_RECORD\_CAPTURE = 0, /\* Recording/snapshot \*/

NETDEV\_STORAGE\_CONTAINER\_USAGE\_TYPE\_BACKUP = 1, /\* Backup \*/

NETDEV\_STORAGE\_CONTAINER\_USAGE\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_STORAGE\_CONTAINER\_USAGE\_TYPE\_E;

### HDD health assessment status

typedef enum tagNETDEVHDDHealthAssessmentStatus

{

NETDEV\_HDD\_HEALTH\_ASSESSMENT\_STATUS\_NORMAL = 0, /\* Normal \*/

NETDEV\_HDD\_HEALTH\_ASSESSMENT\_STATUS\_PART\_DAMAGE = 1, /\* Bad sectors detected \*/

NETDEV\_HDD\_HEALTH\_ASSESSMENT\_STATUS\_FAULT = 2, /\* Fault \*/

NETDEV\_HDD\_HEALTH\_ASSESSMENT\_STATUS\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HDD\_HEALTH\_ASSESSMENT\_STATUS\_E;

### HDD S.M.A.R.T. test status

typedef enum tagNETDEVHDDSmartCheckStatus

{

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_NOT = 0, /\* Not tested \*/

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_IN\_PORGRESS = 1, /\* Testing \*/

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_SUCCESS = 2, /\* Success \*/

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_RECOGNITION\_FAIL = 3, /\* HDD recognition failed \*/

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_FAIL = 4, /\* Failure \*/

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_NOT\_SUPPORT = 5, /\* HDD does not support test \*/

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HDD\_SMART\_CHECK\_STATUS\_E;

### HDD S.M.A.R.T. test types

typedef enum tagNETDEVHDDSmartCheckType

{

NETDEV\_HDD\_SMART\_CHECK\_TYPE\_BRIEF = 0, /\* Short \*/

NETDEV\_HDD\_SMART\_CHECK\_TYPE\_EXTEND = 1, /\* Extended \*/

NETDEV\_HDD\_SMART\_CHECK\_TYPE\_TRANSMISSION = 2, /\* Conveyance \*/

NETDEV\_HDD\_SMART\_CHECK\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HDD\_SMART\_CHECK\_TYPE\_E;

### Access protocols of management server

typedef enum tagNETDEVManagerServerProtocolType

{

NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_IMOS = 0, /\* IMOS \*/

NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_GB = 1, /\* GB\*/

NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_NONE = 2, /\* None \*/

NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_VISS = 3, /\* VISS \*/

NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_LY = 4 /\* LiYuan \*/

}NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_E;

### Storage space unit

typedef enum tagNETDEVStorageCapUnit

{

NETDEV\_TYPE\_STORAGE\_CAP\_UNIT\_GB = 0, /\* GB \*/

NETDEV\_TYPE\_STORAGE\_CAP\_UNIT\_MB = 1, /\* MB \*/

NETDEV\_TYPE\_STORAGE\_CAP\_UNIT\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_STORAGE\_CAP\_UNIT\_E;

### Transport protocol

typedef enum tagNETDEVTransProtocol

{

NETDEV\_TRANS\_PROTOCOL\_TCP = 0, /\* TCP \*/

NETDEV\_TRANS\_PROTOCOL\_UDP = 1 /\* UDP \*/

}NETDEV\_TRANS\_PROTOCOL\_E;

### Storage resources

typedef enum tagNETDEVStorageResType

{

NETDEV\_STORAGE\_RES\_TYPE\_SD = 0, /\* SD card \*/

NETDEV\_STORAGE\_RES\_TYPE\_NAS = 1, /\* NAS \*/

NETDEV\_STORAGE\_RES\_TYPE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_STORAGE\_RES\_TYPE\_E;

### Overwrite policy

typedef enum tagNETDEVFullStrategyType

{

NETDEV\_FULL\_STRATEGY\_TYPE\_FULL\_COVERAGE = 0, /\* Overwrite existing data when storage is full\*/

NETDEV\_FULL\_STRATEGY\_TYPE\_FULL\_STOP = 1, /\* Stop saving data when storage is full \*/

NETDEV\_FULL\_STRATEGY\_TYPE\_INVALIDP = 0xff /\* Invalid value \*/

}NETDEV\_FULL\_STRATEGY\_TYPE\_E;

### Live view index

typedef enum tagNETDEVLiveStreamIndex

{

NETDEV\_LIVE\_STREAM\_INDEX\_MAIN = 0, /\* Main stream \*/

NETDEV\_LIVE\_STREAM\_INDEX\_AUX = 1, /\* Sub stream \*/

NETDEV\_LIVE\_STREAM\_INDEX\_THIRD = 2, /\* Third stream \*/

NETDEV\_LIVE\_STREAM\_INDEX\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_LIVE\_STREAM\_INDEX\_E;

### Image storage modes

typedef enum tagNETDEVPicStorageMode

{

NETDEV\_PIC\_STORAGE\_MODE\_NORMAL = 0, /\* Normal. Delete files in SD card after successful upload to server \*/

NETDEV\_PIC\_STORAGE\_MODE\_PERMANENT = 1, /\* Permanent. Always keep files in SD card after upload to server, successful or not. \*/

NETDEV\_PIC\_STORAGE\_MODE\_RETAIN = 2, /\* Keep files in SD card after upload to server failed. \*/

NETDEV\_PIC\_STORAGE\_MODE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_PIC\_STORAGE\_MODE\_E;

### Back focus control commands

typedef enum tagNETDEVBackFocusCmd

{

NETDEV\_BACKFOCUS\_CMD\_RESET = 0, /\* Automatically reset back focus. Para1 is optional \*/

NETDEV\_BACKFOCUS\_CMD\_AUTO = 1, /\* Automatic back focus. Para1: automatic backfocus time (ms) \*/

NETDEV\_BACKFOCUS\_CMD\_MANUAL = 2, /\* Manual back focus. Para1: 0-Stop 1-MBF+ 2-MBF- \*/

NETDEV\_BACKFOCUS\_CMD\_INVALID = 0xffff/\* Invalid value \*/

}NETDEV\_BACKFOCUS\_CMD\_E;

### Time zone

typedef enum tagNETDEVTimeZone

{

NETDEV\_TIME\_ZONE\_W1200 = 0, /\* W12 \*/

NETDEV\_TIME\_ZONE\_W1100 = 1, /\* W11 \*/

NETDEV\_TIME\_ZONE\_W1000 = 2, /\* W10 \*/

NETDEV\_TIME\_ZONE\_W0900 = 3, /\* W9 \*/

NETDEV\_TIME\_ZONE\_W0800 = 4, /\* W8 \*/

NETDEV\_TIME\_ZONE\_W0700 = 5, /\* W7 \*/

NETDEV\_TIME\_ZONE\_W0600 = 6, /\* W6 \*/

NETDEV\_TIME\_ZONE\_W0500 = 7, /\* W5 \*/

NETDEV\_TIME\_ZONE\_W0430 = 8, /\* W4:30 \*/

NETDEV\_TIME\_ZONE\_W0400 = 9, /\* W4 \*/

NETDEV\_TIME\_ZONE\_W0330 = 10, /\* W3:30 \*/

NETDEV\_TIME\_ZONE\_W0300 = 11, /\* W3 \*/

NETDEV\_TIME\_ZONE\_W0200 = 12, /\* W2 \*/

NETDEV\_TIME\_ZONE\_W0100 = 13, /\* W1 \*/

NETDEV\_TIME\_ZONE\_0000 = 14, /\* W0 \*/

NETDEV\_TIME\_ZONE\_E0100 = 15, /\* E1 \*/

NETDEV\_TIME\_ZONE\_E0200 = 16, /\* E2 \*/

NETDEV\_TIME\_ZONE\_E0300 = 17, /\* E3 \*/

NETDEV\_TIME\_ZONE\_E0330 = 18, /\* E3:30 \*/

NETDEV\_TIME\_ZONE\_E0400 = 19, /\* E4 \*/

NETDEV\_TIME\_ZONE\_E0430 = 20, /\* E4:30 \*/

NETDEV\_TIME\_ZONE\_E0500 = 21, /\* E5 \*/

NETDEV\_TIME\_ZONE\_E0530 = 22, /\* E5:30 \*/

NETDEV\_TIME\_ZONE\_E0545 = 23, /\* E5:45 \*/

NETDEV\_TIME\_ZONE\_E0600 = 24, /\* E6 \*/

NETDEV\_TIME\_ZONE\_E0630 = 25, /\* E6:30 \*/

NETDEV\_TIME\_ZONE\_E0700 = 26, /\* E7 \*/

NETDEV\_TIME\_ZONE\_E0800 = 27, /\* E8 \*/

NETDEV\_TIME\_ZONE\_E0900 = 28, /\* E9 \*/

NETDEV\_TIME\_ZONE\_E0930 = 29, /\* E9:30 \*/

NETDEV\_TIME\_ZONE\_E1000 = 30, /\* E10 \*/

NETDEV\_TIME\_ZONE\_E1100 = 31, /\* E11 \*/

NETDEV\_TIME\_ZONE\_E1200 = 32, /\* E12 \*/

NETDEV\_TIME\_ZONE\_E1300 = 33, /\* E13 \*/

NETDEV\_TIME\_ZONE\_W0930= 34, /\* W9:30 \*/

NETDEV\_TIME\_ZONE\_E0830 = 35, /\* E8:30 \*/

NETDEV\_TIME\_ZONE\_E0845 = 36, /\* E8:45 \*/

NETDEV\_TIME\_ZONE\_E1030 = 37, /\* E10:30 \*/

NETDEV\_TIME\_ZONE\_E1245 = 38, /\* E12:45 \*/

NETDEV\_TIME\_ZONE\_E1400 = 39, /\* E14 \*/

NETDEV\_TIME\_ZONE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_TIME\_ZONE\_E;

### Illuminator control modes

typedef enum tagNETDEVLampCtrlMode

{

NETDEV\_LCM\_GLOBAL\_ATTENTION\_AC = 0, /\* Global mode \*/

NETDEV\_LCM\_OVEREXPOSE\_INHIBIT\_AC = 1, /\* Overexposure restrain \*/

NETDEV\_LCM\_PRESET\_RCM = 2, /\* Preset-road mode \*/

NETDEV\_LCM\_MANUAL\_CM = 3, /\* Manual mode \*/

NETDEV\_LCM\_PRESET\_PCM = 4, /\* Preset-park mode\*/

NETDEV\_LCM\_PHOTOSENS\_AM = 5, /\* Auto sensitivity control mode \*/

NETDEV\_LCM\_INDOOR\_AM = 6, /\* Indoor \*/

NETDEV\_LCM\_MANUAL\_FOM = 7, /\* Manual-always on\*/

NETDEV\_LCM\_NIGHT\_VIDEO\_DET\_MODE = 8, /\* Nighttime video detection mode \*/

NETDEV\_LCM\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_LAMP\_CTRL\_MODE\_E;

### HDD S.M.A.R.T. test result

typedef enum tagNETDEVHDDSmartAssessmentStatus

{

NETDEV\_HDD\_SMART\_ASSESSMENT\_STATUS\_NORMAL = 0, /\* Normal \*/

NETDEV\_HDD\_SMART\_ASSESSMENT\_STATUS\_WARNING = 1, /\* Warning \*/

NETDEV\_HDD\_SMART\_ASSESSMENT\_STATUS\_FAULT = 2, /\* Fault \*/

NETDEV\_HDD\_SMART\_ASSESSMENT\_STATUS\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HDD\_SMART\_ASSESSMENT\_STATUS\_E;

### Communication protocol of photo server

typedef enum tagNETDEVPhotoServerProType

{

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO1 = 0, /\* TCP transport: Private V1 (corresponds to HongLiu) \*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO2 = 1, /\* TCP transport: Private V2 (corresponds to current tms）\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO3 = 2, /\* TCP transport: XianChuang\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO4 = 3, /\* FTP transport: QuanShiTong\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO5 = 4, /\* FTP transport: General FTP \*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO6 = 5, /\* FTP transport: ZhengZhi\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO7 = 6, /\* webserver: Hikvision protocol \*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO8 = 7, /\* FTP transport: ZhongTong\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO9 = 8, /\* FTP transport: QuanShiTong\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO10 = 9, /\* YinJiang, photo info: webserver, photo data: FTP transport \*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO11 = 10, /\* TCP transport: LiYuan \*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO12 = 11, /\* XiangXun, photo info: TCP, photo data: FTP transport \*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO31 = 12, /\* TCP transport: V2 (corresponds to current TMS non-persistent connection)\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO34 = 13, /\* HTTP transport: SongLi\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_E;

### Sub communication protocols of photo server

typedef enum tagNETDEVPhotoServerSubProType

{

NETDEV\_PHOTO\_SERVER\_SUBPRO\_TYPE\_2011 = 0, /\* IMOS command\*/

NETDEV\_PHOTO\_SERVER\_SUBPRO\_TYPE\_2025 = 1, /\* IMOS command\*/

NETDEV\_PHOTO\_SERVER\_SUBPRO\_TYPE\_6130 = 2, /\* IMOS command\*/

NETDEV\_PHOTO\_SERVER\_SUBPRO\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_PHOTO\_SERVER\_SUBPRO\_TYPE\_E;

### Access protocols of smart server

typedef enum tagNETDEVDataServerProtocolType

{

NETDEV\_DATA\_SERVER\_PROTOCOL\_TYPE\_TCP = 1, /\* TCP transport: (corresponds to tms) \*/

NETDEV\_DATA\_SERVER\_PROTOCOL\_TYPE\_TFTP\_COMMOM = 4, /\* FTP transport: General FPT \*/

NETDEV\_DATA\_SERVER\_PROTOCOL\_TYPE\_HTTP\_GA = 51 /\* HTTP transport: VIID \*/

}NETDEV\_DATA\_SERVER\_PROTOCOL\_TYPE\_E;

### OSD content type

typedef enum tagNETDEVOSDContentType

{

NETDEV\_OSD\_CONTENT\_TYPE\_NOTUSE = 0, /\* Not use \*/

NETDEV\_OSD\_CONTENT\_TYPE\_CUSTOM = 1, /\* Custom \*/

NETDEV\_OSD\_CONTENT\_TYPE\_DATE\_AND\_TIME = 2, /\* Time and date \*/

NETDEV\_OSD\_CONTENT\_TYPE\_PTZ\_CONTROLLER = 3, /\* PTZ controller \*/

NETDEV\_OSD\_CONTENT\_TYPE\_PTZ\_COORDINATES = 4, /\* PTZ coordinates \*/

NETDEV\_OSD\_CONTENT\_TYPE\_CRUISE = 5, /\* Patrol info \*/

NETDEV\_OSD\_CONTENT\_TYPE\_ZOOM = 6, /\* Zoom info \*/

NETDEV\_OSD\_CONTENT\_TYPE\_PRESET = 7, /\* Preset info \*/

NETDEV\_OSD\_CONTENT\_TYPE\_ALARM\_INFO = 8, /\* Alarm info \*/

NETDEV\_OSD\_CONTENT\_TYPE\_ENCODE = 9, /\* Encoding info \*/

NETDEV\_OSD\_CONTENT\_TYPE\_SERIAL\_PORT = 10, /\* Serial port OSD \*/

NETDEV\_OSD\_CONTENT\_TYPE\_PTZ\_ORIENTATION = 11, /\* PTZ direction info\*/

NETDEV\_OSD\_CONTENT\_TYPE\_CHN\_NAME = 12, /\* Channel name \*/

NETDEV\_OSD\_CONTENT\_TYPE\_DEBUG = 13, /\* Debug OSD \*/

NETDEV\_OSD\_CONTENT\_TYPE\_PEOPLE\_COUNTING = 14, /\* People counting OSD \*/

NETDEV\_OSD\_CONTENT\_TYPE\_NETWORK\_PORT = 15, /\* Network port OSD \*/

NETDEV\_OSD\_CONTENT\_TYPE\_TIME = 16, /\* Time \*/

NETDEV\_OSD\_CONTENT\_TYPE\_DATE = 17, /\* Date \*/

NETDEV\_OSD\_CONTENT\_TYPE\_SMART\_CONTENT = 18, /\* Smart contents \*/

NETDEV\_OSD\_CONTENT\_TYPE\_BATTERY = 19, /\* Battery \*/

NETDEV\_OSD\_CONTENT\_TYPE\_SCROLL\_OSD = 20, /\* Scrolling OSD \*/

NETDEV\_OSD\_CONTENT\_TYPE\_PICTURE\_OVERLAY = 21, /\* Logo \*/

NETDEV\_OSD\_CONTENT\_TYPE\_MOTOR\_VEHICLE\_NUM = 22, /\* Motor vehicle traffic \*/

NETDEV\_OSD\_CONTENT\_TYPE\_NON\_MOTOR\_VEHICLE\_NUM = 23, /\* Non-motor vehicle traffic \*/

NETDEV\_OSD\_CONTENT\_TYPE\_PEOPLE\_NUM = 24, /\* Pedestrian flow \*/

NETDEV\_OSD\_CONTENT\_TYPE\_INFOOSD\_NUM = 25 /\* Number of INFO OSD types \*/

}NETDEV\_OSD\_CONTENT\_TYPE\_E;

### Enumeration of device types

typedef enum tagNETDEVDeviceMainType

{

NETDEV\_DTYPE\_MAIN\_ENCODE = 0, /\* Encoding device \*/

NETDEV\_DTYPE\_MAIN\_DECODE = 1, /\* Decoding device \*/

NETDEV\_DTYPE\_MAIN\_VMS = 2, /\* VMS device\*/

NETDEV\_DTYPE\_MAIN\_DA = 3, /\* Device agent\*/

NETDEV\_DTYPE\_MAIN\_CLOUD = 4, /\* Cloud encoding device \*/

NETDEV\_DTYPE\_MAIN\_BAYONET = 5, /\* Checkpoint device \*/

NETDEV\_DTYPE\_MAIN\_ACS = 6, /\* Access control device \*/

NETDEV\_DTYPE\_MAIN\_ALARMHOST = 7, /\* Alarm control device \*/

NETDEV\_DTYPE\_MAIN\_EXITENTRANCE = 8, /\* Entrance&exit device \*/

NETDEV\_DTYPE\_MAIN\_LOCK = 9, /\* Lock \*/

NETDEV\_DTYPE\_MAIN\_PLAYER\_MANAGER = 10, /\* Player manager \*/

NETDEV\_DTYPE\_MAIN\_PLAYER = 11, /\* Player \*/

NETDEV\_DTYPE\_MAIN\_UNKNOWN = 0XFF /\* Unknown device \*/

}NETDEV\_DEVICE\_MAIN\_TYPE\_E;

### IP address types

typedef enum tagNETSDKIpAddressType

{

NETDEV\_IP\_TYPE\_IPV4 = 0, /\* IPv4 \*/

NETDEV\_IP\_TYPE\_IPV6 = 1, /\* IPv6 \*/

NETDEV\_IP\_TYPE\_DOMAIN = 2, /\* Domain name\*/

NETDEV\_IP\_TYPE\_IPV4\_AND\_IPV6 = 3 /\* IPv4 and IPv6 \*/

}NETDEV\_IP\_ADDRESS\_TYPE\_E;

### Lock signals

typedef enum tagNETDEVLockSignal

{

NETDEV\_LOCK\_SIGNAL\_STRONG = 0, /\* Strong \*/

NETDEV\_LOCK\_SIGNAL\_SECOND = 1, /\* Medium \*/

NETDEV\_LOCK\_SIGNAL\_WEAK = 2, /\* Weak \*/

NETDEV\_LOCK\_SIGNAL\_NO = 3, /\* No signal \*/

NETDEV\_LOCK\_SIGNAL\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_LOCK\_SIGNAL\_E;

### Channel type

typedef enum tagNETDEVChlType

{

NETDEV\_CHN\_TYPE\_ENCODE = 0, /\* Encoding channel. For channel status, see

[NETDEV\_CHN\_STATUS\_E](#_编码通道状态). To get data, see NETDEV\_DEV\_CHN\_ENCODE\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_ALARMIN = 1, /\* Alarm input channel. For channel status, see

NETDEV\_ALARM\_RUNMODE\_E. To get data, see NETDEV\_DEV\_CHN\_ALARMIN\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_ALARMOUT = 2, /\* Alarm output channel. For channel status, see

NETDEV\_ALARMOUT\_CHN\_STATUS\_E. To get data, see NETDEV\_DEV\_CHN\_ALARMOUT\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_DECODE = 3, /\* Decoding channel. To get data, see NETDEV\_DEV\_CHN\_DECODE\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_AUDIO = 4, /\* Audio channel. To get data, see NETDEV\_DEV\_CHN\_AUDIO\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_NIC = 5, /\* Network interface card \*/

NETDEV\_CHN\_TYPE\_ALARMPOINT = 6, /\* Alarm point. To get data, see NETDEV\_DEV\_CHN\_DA\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_DOOR = 7, /\* Door. To get data, see NETDEV\_DEV\_CHN\_DA\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_ADU\_ENCODE = 8, /\* ADU’s local encoding channel. For channel status, see

[NETDEV\_CHN\_STATUS\_E](#_编码通道状态). To get data, see #NETDEV\_DEV\_CHN\_XW\_ENCODE\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_EMERGENCY = 9, /\* Emergency bell. To get data, see NETDEV\_EMERGENCY\_BELL\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_CHN\_TYPE\_E;

### Status of encoding channel

typedef enum tagNETDEVChnStatus

{

NETDEV\_CHN\_STATUS\_OFFLINE = 0, /\* Offline \*/

NETDEV\_CHN\_STATUS\_ONLINE = 1, /\* Online \*/

NETDEV\_CHN\_STATUS\_VIDEO\_LOSE = 2, /\* Video loss \*/

NETDEV\_CHN\_STATUS\_INVALID = 0xFF

}NETDEV\_CHN\_STATUS\_E;

### Alarm input channel status

typedef enum tagNETDEVAlarmRunode

{

NETDEV\_ALARM\_RUNMODE\_OPEN = 0, /\* Normally open \*/

NETDEV\_ALARM\_RUNMODE\_CLOSE = 1, /\* Normally closed \*/

NETDEV\_ALARM\_RUNMODE\_INVALID = 0xFF

}NETDEV\_ALARM\_RUNMODE\_E;

### Alarm output channel status

typedef enum tagNETDEVAlarmOutChnStatus

{

NETDEV\_ALARMOUT\_CHN\_ACTIVE = 0,

NETDEV\_ALARMOUT\_CHN\_INACTIVE = 1,

NETDEV\_ALARMOUT\_CHN\_INVALID = 0xFF

}NETDEV\_ALARMOUT\_CHN\_STATUS\_E;

### Enumeration of video input types

typedef enum tagNETDEVVideoSrcType

{

NETDEV\_VIDEO\_SRC\_TYPE\_BNC = 0, /\* BNC \*/

NETDEV\_VIDEO\_SRC\_TYPE\_VGA = 1, /\* VGA \*/

NETDEV\_VIDEO\_SRC\_TYPE\_HDMI = 2, /\* HDMI \*/

NETDEV\_VIDEO\_SRC\_TYPE\_DVI\_D = 3, /\* DVI\_D \*/

NETDEV\_VIDEO\_SRC\_TYPE\_DVI\_I = 4, /\* DVI\_I \*/

NETDEV\_VIDEO\_SRC\_TYPE\_DP = 5, /\* DP \*/

NETDEV\_VIDEO\_SRC\_TYPE\_SDI = 6, /\* SDI \*/

NETDEV\_VIDEO\_SRC\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_VIDEO\_SRC\_TYPE\_E;

### Enumeration of language types

typedef enum tagNETDEVLangType

{

NETDEV\_LANG\_TYPE\_CHINESE = 0, /\* Chinese \*/

NETDEV\_LANG\_TYPE\_ENGLISH = 1, /\* English \*/

NETDEV\_LANG\_TYPE\_TRADITIONAL\_CHINESE = 2, /\* Traditional Chinese \*/

NETDEV\_LANG\_TYPE\_FRENCH = 3, /\* French \*/

NETDEV\_LANG\_TYPE\_GERMAN = 4, /\* German \*/

NETDEV\_LANG\_TYPE\_SPANISH = 5, /\* Spanish \*/

NETDEV\_LANG\_TYPE\_ITALIAN = 6, /\* Italian \*/

NETDEV\_LANG\_TYPE\_PORTUGUESE = 7, /\* Portuguese \*/

NETDEV\_LANG\_TYPE\_RUSSIAN = 8, /\* Russian \*/

NETDEV\_LANG\_TYPE\_UKRAINIAN = 9, /\* Ukrainian \*/

NETDEV\_LANG\_TYPE\_GRUANIAN = 10, /\* Gruanian \*/

NETDEV\_LANG\_TYPE\_DUTCH = 11, /\* Dutch \*/

NETDEV\_LANG\_TYPE\_DANISH = 12, /\* Danish \*/

NETDEV\_LANG\_TYPE\_FINNISH = 13, /\* Finnish \*/

NETDEV\_LANG\_TYPE\_SWEDISH = 14, /\* Swedish \*/

NETDEV\_LANG\_TYPE\_NORWEGIAN = 15, /\* Norwegian \*/

NETDEV\_LANG\_TYPE\_LCELANDIC = 16, /\* Icelandic \*/

NETDEV\_LANG\_TYPE\_POLISH = 17, /\* Polish \*/

NETDEV\_LANG\_TYPE\_LITHUANIAN = 18, /\* Lithuanian \*/

NETDEV\_LANG\_TYPE\_GREEK = 19, /\* Greek \*/

NETDEV\_LANG\_TYPE\_CZECH = 20, /\* Czech \*/

NETDEV\_LANG\_TYPE\_SLOVAK = 21, /\* Slovak \*/

NETDEV\_LANG\_TYPE\_HUNGARIAN = 22, /\* Hungarian \*/

NETDEV\_LANG\_TYPE\_ROMAN = 23, /\* Roman \*/

NETDEV\_LANG\_TYPE\_SERBIAN = 24, /\* Serbian \*/

NETDEV\_LANG\_TYPE\_CROATIAN = 25, /\* Croatian \*/

NETDEV\_LANG\_TYPE\_SLOVENIAN = 26, /\* Slovenian \*/

NETDEV\_LANG\_TYPE\_BULGARIAN = 27, /\* Bulgarian \*/

NETDEV\_LANG\_TYPE\_BELARUSIAN = 28, /\* Belarusian \*/

NETDEV\_LANG\_TYPE\_TURKISH = 29, /\* Turkish \*/

NETDEV\_LANG\_TYPE\_ARABIC = 30, /\* Arabic \*/

NETDEV\_LANG\_TYPE\_HEBREW = 31, /\* Hebrew \*/

NETDEV\_LANG\_TYPE\_HINDI = 32, /\* Hindi \*/

NETDEV\_LANG\_TYPE\_BENGALI = 33, /\* Bengali \*/

NETDEV\_LANG\_TYPE\_PERSIAN = 34, /\* Persian \*/

NETDEV\_LANG\_TYPE\_JAPANESE = 35, /\* Japanese \*/

NETDEV\_LANG\_TYPE\_KOREAN = 36, /\* Korean\*/

NETDEV\_LANG\_TYPE\_THAI = 37, /\* Thai \*/

NETDEV\_LANG\_TYPE\_MALAYSIAN = 38, /\* Malaysian \*/

NETDEV\_LANG\_TYPE\_INDONESIAN = 39, /\* Indonesian \*/

NETDEV\_LANG\_TYPE\_FILIPINO = 40, /\* Filipino \*/

NETDEV\_LANG\_TYPE\_LAO = 41, /\* Lao \*/

NETDEV\_LANG\_TYPE\_VIETNAMESE = 42, /\* Vietnamese \*/

NETDEV\_LANG\_TYPE\_MONGOLIAN = 43, /\* Mongolian \*/

NETDEV\_LANG\_TYPE\_ALBANIAN = 44, /\* Albanian \*/

NETDEV\_LANG\_TYPE\_AZERBAIJANI = 45, /\* Azerbaijani \*/

NETDEV\_LANG\_TYPE\_GAELIC = 46, /\* Gaelic \*/

NETDEV\_LANG\_TYPE\_ESTONIAN = 47, /\* Estonian \*/

NETDEV\_LANG\_TYPE\_BOSNIAN = 48, /\* Bosnian \*/

NETDEV\_LANG\_TYPE\_LOWLAND\_SAXON = 49, /\* Lowland Saxon \*/

NETDEV\_LANG\_TYPE\_PERUVIAN = 50, /\* Peruvian \*/

NETDEV\_LANG\_TYPE\_LATVIAN = 51, /\* Latvian \*/

NETDEV\_LANG\_TYPE\_INVALID = 0xFF /\* Invalid \*/

}NETDEV\_LANG\_TYPE\_E;

### Enumeration of client types

typedef enum tagClientType

{

NETDEV\_CLIENT\_TYPE\_EZVMS = 0, /\* EZVMS \*/

NETDEV\_CLIENT\_TYPE\_EZGATE = 1, /\* EZGate \*/

NETDEV\_CLIENT\_TYPE\_EZASSIST = 2, /\* EZAssist \*/

NETDEV\_CLIENT\_TYPE\_IVALID = 0xff /\* Invalid value \*/

}NETDEV\_CLIENT\_TYPE\_E;

### Enumeration of channel statuses

typedef enum tagNETDEVChannelStatus

{

NETDEV\_CHL\_STATUS\_OFFLINE = 0, /\* Offline \*/

NETDEV\_CHL\_STATUS\_ONLINE = 1, /\* Online \*/

NETDEV\_CHL\_STATUS\_UNBIND = 2, /\* Idle \*/

NETDEV\_CHL\_STATUS\_INVALID

}NETDEV\_CHANNEL\_STATUS\_E;

### Channel type

typedef enum tagNETDEVChannelType

{

NETDEV\_CHL\_TYPE\_DIGITAL = 0, /\* Digital \*/

NETDEV\_CHL\_TYPE\_ANALOG = 1, /\* Analog \*/

NETDEV\_CHL\_TYPE\_INVALID

}NETDEV\_CHANNEL\_TYPE\_E;

### Video standard

typedef enum tagNETDEVVideoStandard

{

NETDEV\_VIDEO\_STANDARD\_PAL = 0, /\* PAL \*/

NETDEV\_VIDEO\_STANDARD\_NTSC = 1, /\* NTSC \*/

NETDEV\_VIDEO\_STANDARD\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_VIDEO\_STANDARD\_E;

### Enumeration of PoE port status

typedef enum tagNETDEVPoEStatus

{

NETDEV\_POE\_STATUS\_NORMAL = 1,

NETDEV\_POE\_STATUS\_POWER\_SHORTAGE = 2,

NETDEV\_POE\_STATUS\_POWER\_OVERLOAP = 4,

NETDEV\_POE\_STATUS\_NONEED\_POWER = 8,

NETDEV\_POE\_STATUS\_INVALID = 0xFF

}NETDEV\_POE\_STATUS\_E;

### Enumeration of device offline causes

typedef enum tagNETDEVChnOffReason

{

NETDEV\_CHN\_OFF\_REASON\_CONNECTING = 0, /\* Connecting \*/

NETDEV\_CHN\_OFF\_REASON\_ONLINE = 1, /\* Online \*/

NETDEV\_CHN\_OFF\_REASON\_WRONG\_PWD = 2, /\* Incorrect username or password \*/

NETDEV\_CHN\_OFF\_REASON\_NETWORK\_ERROR = 3, /\* Network error \*/

NETDEV\_CHN\_OFF\_REASON\_SET\_LIVE\_STREAM\_ERROR = 4, /\* Failed to set live stream \*/

NETDEV\_CHN\_OFF\_REASON\_START\_LIVE\_STREAM\_ERROR = 5, /\* Failed to start live stream \*/

NETDEV\_CHN\_OFF\_REASON\_STREAM\_INTERUP = 6, /\* Media stream interrupted \*/

NETDEV\_CHN\_OFF\_REASON\_TIMEOUT = 7, /\* Timeout \*/

NETDEV\_CHN\_OFF\_REASON\_WAIT\_REGIST = 8, /\* GB IPC waiting for registration packets \*/

NETDEV\_CHN\_OFF\_REASON\_MODIFY\_LIVE\_STREAM\_ERROR = 9, /\* GB IPC modify stream transport protocol \*/

NETDEV\_CHN\_OFF\_REASON\_POE\_POWER\_ERROR = 10, /\* PoE power error \*/

NETDEV\_CHN\_OFF\_REASON\_BW\_RECV\_NOT\_ENOUGH = 11, /\* Bandwidth not enough \*/

NETDEV\_CHN\_OFF\_REASON\_WEAK\_PWD\_ERROR = 12, /\* Access denied for weak password \*/

NETDEV\_CHN\_OFF\_REASON\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_CHN\_OFF\_REASON\_E;

### Device status

typedef enum tagNETDEVDeviceStatus

{

NETDEV\_DEV\_STATUS\_OFFLINE = 0, /\* Offline \*/

NETDEV\_DEV\_STATUS\_ONLINE = 1, /\* Online \*/

NETDEV\_DEV\_STATUS\_CONNECTING = 2, /\* Connecting \*/

NETDEV\_DEV\_STATUS\_PWD\_ERROR = 3, /\* Incorrect username or password \*/

NETDEV\_DEV\_STATUS\_NOT\_SUPPORT = 4, /\* Not supported by device \*/

NETDEV\_DEV\_STATUS\_TIMEOUT = 5, /\* Device connection timed out \*/

NETDEV\_DEV\_STATUS\_WEAK\_PWD\_ERROR = 6, /\* Weak password error for remote user \*/

NETDEV\_DEV\_STATUS\_NO\_DYNAMIC\_PWD = 7, /\* Device does not support dynamic password \*/

NETDEV\_DEV\_STATUS\_INVALID = 0XFF /\* Invalid value \*/

}NETDEV\_DEVICE\_STATUS\_E;

### Device sub types

typedef enum tagNETDEVDeviceSubType

{

NETDEV\_DTYPE\_SUB\_NVR = 0, /\* NVR \*/

NETDEV\_DTYPE\_SUB\_IPC = 1, /\* IPC \*/

NETDEV\_DTYPE\_SUB\_DC\_INNER = 2, /\* Built-in DC \*/

NETDEV\_DTYPE\_SUB\_DC\_EXT = 3, /\* External DC \*/

NETDEV\_DTYPE\_SUB\_EC = 4, /\* EC \*/

NETDEV\_DTYPE\_SUB\_VMS = 5, /\* VMS \*/

NETDEV\_DTYPE\_SUB\_DA = 6, /\* Device agent \*/

NETDEV\_DTYPE\_SUB\_ADU = 7, /\* ADU \*/

NETDEV\_DTYPE\_SUB\_FISHEYE\_IPC = 8, /\* Fisheye device \*/

NETDEV\_DTYPE\_SUB\_DECARD = 9, /\* Decoding card \*/

NETDEV\_DTYPE\_SUB\_FACE\_IPC = 10, /\* Face recognition camera \*/

NETDEV\_DTYPE\_SUB\_ALARMHOST = 11, /\* Alarm control device \*/

NETDEV\_DTYPE\_SUB\_ACS = 12, /\* 3rd-party access control device \*/

NETDEV\_DTYPE\_SUB\_FG = 13, /\* Face recognition terminal \*/

NETDEV\_DTYPE\_SUB\_FACE\_ACS = 14, /\* Face recognition access control device \*/

NETDEV\_DTYPE\_SUB\_PRIVATE\_GENERAL\_ACS = 15, /\* General access control device \*/

NETDEV\_DTYPE\_SUB\_PRIVATE\_ALARMHOST = 16, /\* Alarm control device \*/

NETDEV\_DTYPE\_SUB\_LOCK\_WIF = 17, /\* Wi-Fi lock\*/

NETDEV\_DTYPE\_SUB\_LOCK\_NBIoT = 18, /\* NBIoT lock\*/

NETDEV\_DTYPE\_SUB\_ID\_RECOGNIZE = 19, /\* Face&ID recognition device \*/

NETDEV\_DTYPE\_SUB\_UNKNOW = 0xFF /\* Unknown device \*/

}NETDEV\_DEVICE\_SUB\_TYPE\_E;

### Access protocol

typedef enum tagNETDEVAccessProtocol

{

NETDEV\_ACCESS\_PROTOCOL\_ONVIF = 1, /\* Onvif \*/

NETDEV\_ACCESS\_PROTOCOL\_PRIVATE = 2, /\* Private \*/

NETDEV\_ACCESS\_PROTOCOL\_GB = 3, /\* GB/T28181\*/

NETDEV\_ACCESS\_PROTOCOL\_RTSP = 4, /\* RTSP \*/

NETDEV\_ACCESS\_PROTOCOL\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_ACCESS\_PROTOCOL\_E;

### Access mode

typedef enum tagNETDEVDeviceAccessMode

{

NETDEV\_DEV\_ACCESS\_MODE\_IP\_DOMAIN = 0, /\* IP, domain name \*/

NETDEV\_DEV\_ACCESS\_MODE\_P2P = 1, /\* P2P \*/

NETDEV\_DEV\_ACCESS\_MODE\_UNP = 2, /\* UNP \*/

NETDEV\_DEV\_ACCESS\_MODE\_INVALID = 0XFF /\* Invalid value \*/

}NETDEV\_DEVICE\_ACCESS\_MODE\_E;

### Types of devices connected to channels (for NVR only)

typedef enum tagNETDEVChannelCameraType

{

NETDEV\_CHL\_CAMERA\_TYPE\_FIXED = 0, /\* Fixed camera \*/

NETDEV\_CHL\_CAMERA\_TYPE\_PTZ = 1, /\* PTZ camera \*/

NETDEV\_CHL\_CAMERA\_TYPE\_FISH\_EYE = 2, /\* Fisheye (fisheye lens + dewarping functionality) \*/

NETDEV\_CHL\_CAMERA\_TYPE\_WIDE\_ANGLE = 3, /\* Wide-angle lens camera (including fisheye or wide-angle lens but without dewarping functionality) \*/

NETDEV\_CHL\_CAMERA\_TYPE\_INVALID = 0xff

}NETDEV\_CHANNEL\_CAMERA\_TYPE\_E;

### Alarm/event report type

typedef enum tagNETDEVReportType

{

NETDEV\_REPORT\_TYPE\_ALARM = 0, /\* Type: Alarm \*/

NETDEV\_REPORT\_TYPE\_EVENT = 1, /\* Type: Event \*/

NETDEV\_REPORT\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_REPORT\_TYPE\_E;

### Alarm resource types

typedef enum tagNETDEVAlarmSrcTypeEn

{

NETDEV\_ALARM\_SRC\_LOCAL\_HARD\_DISK = 0, /\* Local disk \*/

NETDEV\_ALARM\_SRC\_EX\_CABINET\_1\_STORAGE\_DISK = 1, /\* Expansion enclosure-1 disk\*/

NETDEV\_ALARM\_SRC\_EX\_CABINET\_2\_STORAGE\_DISK = 2, /\* Expansion enclosure-2 disk \*/

NETDEV\_ALARM\_SRC\_ARRAY\_STORAGE\_DISK = 3, /\* Array disk \*/

NETDEV\_ALARM\_SRC\_ARRAY\_STORAGE\_DISK = 4, /\* NAS disk \*/

NETDEV\_ALARM\_SRC\_ARRAY\_STORAGE\_DISK = 5, /\* Array disk \*/

NETDEV\_ALARM\_SRC\_ESATA\_STORAGE\_DISK = 6, /\*eSATA disk \*/

NETDEV\_ALARM\_SRC\_SD\_STORAGE\_DISK = 7, /\*SD card \*/

NETDEV\_ALARM\_SRC\_VIDEO\_CHANNEL = 8, /\*Video channel \*/

NETDEV\_ALARM\_SRC\_ALARM\_INPUT\_CHANNEL = 9, /\* Alarm input channel \*/

NETDEV\_ALARM\_SRC\_SYSTEM = 10, /\* System \*/

NETDEV\_ALARM\_SRC\_AUDIO\_CHANNEL = 11, /\*Audio channel \*/

NETDEV\_ALARM\_SRC\_DECODE\_CHANNEL = 12, /\*Decoding channel \*/

NETDEV\_ALARM\_SRC\_ALARM\_OUTPUT\_CHANNEL = 13, /\*Alarm output channel \*/

NETDEV\_ALARM\_SRC\_ACCESS\_CONTROL\_CHANNEL = 14, /\* Access control channel \*/

NETDEV\_ALARM\_SRC\_ALARM\_POINT\_CHANNEL = 15, /\* Alarm point channel \*/

NETDEV\_ALARM\_SRC\_EMERGENCY\_BELL\_CHANNEL = 16, /\* Emergency bell channel (third-party alarm) \*/

NETDEV\_ALARM\_SRC\_DEV = 17, /\* Device \*/

NETDEV\_ALARM\_SRC\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_ALARM\_SRC\_TYPE\_E;

### Enumerations of media types

typedef enum tagNetSDKMediaMode

{

NETDEV\_MEDIA\_MODE\_LOCAL = 0, /\* Local file \*/

NETDEV\_MEDIA\_MODE\_REALPLAY = 1, /\* Real-time network stream \*/

NETDEV\_MEDIA\_MODE\_PLAYBACK = 2, /\* Network stream playback \*/

NETDEV\_MEDIA\_MODE\_DOWNLOAD = 3, /\* Network stream download \*/

NETDEV\_MEDIA\_MODE\_VOICECOM = 4, /\* Two-way audio \*/

NETDEV\_MEDIA\_MODE\_VOICESRV = 5, /\* Audio data service \*/

NETDEV\_MEDIA\_MODE\_M3U8PLAY = 6, /\* M3u8 file playing \*/

NETDEV\_MEDIA\_MODE\_UNKNOWN = 0xff /\* Unknown mode \*/

}NETDEV\_MEDIA\_MODE\_E;

### Enumeration of types of operation info reported by the decoding layer

typedef enum tagNETDEVPlayerRunInfoType

{

NETDEV\_PLAYER\_RUN\_INFO\_RECORD\_VIDEO = 1, /\*\*< Operation info reported during local recording \*/

NETDEV\_PLAYER\_RUN\_INFO\_MEDIA\_PROCESS = 2, /\*\*< Operation info reported during video media processing \*/

NETDEV\_PLAYER\_RUN\_INFO\_SERIES\_SNATCH = 3, /\*\*< Operation info reported during continuous snapshots \*/

NETDEV\_PLAYER\_RUN\_INFO\_MEDIA\_VOICE = 4, /\*\*< Operation info reported during audio service \*/

NETDEV\_PLAYER\_RUN\_INFO\_MEDIA\_NOT\_IDENTIFY = 5, /\*\*< Unidentified stream \*/

NETDEV\_PLAYER\_RUN\_INFO\_RECV\_PACKET\_NUM = 6, /\*\*< Number of packets received during the period \*/

NETDEV\_PLAYER\_RUN\_INFO\_RECV\_BYTE\_NUM = 7, /\*\*< Number of bytes received during the period \*/

NETDEV\_PLAYER\_RUN\_INFO\_VIDEO\_FRAME\_NUM = 8, /\*\*< Number of video frames decoded during the period \*/

NETDEV\_PLAYER\_RUN\_INFO\_AUDIO\_FRAME\_NUM = 9, /\*\*< Number of audio frames decoded during the period \*/

NETDEV\_PLAYER\_RUN\_INFO\_LOST\_PACKET\_RATIO = 10, /\*\*< Packet loss rate during the period (unit: .01%)\*/

NETDEV\_PLAYER\_RUN\_INFO\_MEDIA\_PLAY\_PROGRESS = 11, /\*\*< Progress info carried in the media \*/

NETDEV\_PLAYER\_RUN\_INFO\_MEDIA\_PLAY\_END = 12, /\*\*< Play end carried in the media \*/

NETDEV\_PLAYER\_RUN\_INFO\_MEDIA\_ABNORMAL = 13, /\*\*< Media processing abnormal \*/

NETDEV\_PLAYER\_RUN\_INFO\_CODEC = 14, /\*\*< Codec \*/

NETDEV\_PLAYER\_RUN\_INFO\_STREAM = 15, /\*\*< Network stream or input stream playing \*/

NETDEV\_PLAYER\_RUN\_INFO\_PLAYBACK\_FINISH = 16, /\*\*< Playback end \*/

NETDEV\_PLAYER\_RUN\_INFO\_SNATCH = 17, /\*\*< Operation info reported during snapshot \*/

NETDEV\_PLAYER\_RUN\_INFO\_INVALID = 0xff

}NETDEV\_PLAYER\_RUN\_INFO\_TYPE\_E;

### Enumeration of event action types

typedef enum tagNETDEVEventActionType

{

NETDEV\_EVENT\_ACTION\_TYPE\_ADD = 0, /\* Event action type: add \*/

NETDEV\_EVENT\_ACTION\_TYPE\_DELETE = 1, /\* Event action type: delete \*/

NETDEV\_EVENT\_ACTION\_TYPE\_MODIFY = 2, /\* Event action type: modify \*/

NETDEV\_EVENT\_ACTION\_TYPE\_ONLINE = 3, /\* Event action type: online \*/

NETDEV\_EVENT\_ACTION\_TYPE\_OFFLINE = 4, /\* Event action type: offline \*/

NETDEV\_EVENT\_ACTION\_TYPE\_EMAP\_ALARM = 5, /\* Event action type: e-map alarm \*/

NETDEV\_EVENT\_ACTION\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_EVENT\_ACTION\_TYPE\_E;

### Enumeration of event resources types

typedef enum tagNETDEVEventResType

{

NETDEV\_EVENT\_RES\_TYPE\_USER = 0, /\* User resource, corresponds to the login handle for user online/offline, otherwise, corresponds to user ID \*/

NETDEV\_EVENT\_RES\_TYPE\_DEVICE = 1, /\* Device resource, corresponds to device ID \*/

NETDEV\_EVENT\_RES\_TYPE\_CHANNEL = 2, /\* Channel resource, corresponds to channel ID \*/

NETDEV\_EVENT\_RES\_TYPE\_LOGOUT = 3, /\* Forced user logout, corresponds to login ID \*/

NETDEV\_EVENT\_RES\_TYPE\_SEQUENCE = 4, /\* Sequence resource, corresponds to resource ID \*/

NETDEV\_EVENT\_RES\_TYPE\_EMAP\_HOTPO = 5, /\* E-map hotspot resource, corresponds to resource ID \*/

NETDEV\_EVENT\_RES\_TYPE\_EMAP\_HOTAREA = 6, /\* E-map hot zone resource, corresponds to resource ID \*/

NETDEV\_EVENT\_RES\_TYPE\_EMAP\_ALARM = 7, /\* E-map alarm resource, corresponds to resource ID \*/

NETDEV\_EVENT\_RES\_TYPE\_TIMETEMPLATE = 8, /\* Alarm plan template, corresponds to template ID \*/

NETDEV\_EVENT\_RES\_TYPE\_SYSRIGH = 9, /\* System permission resource, corresponds to user login handle \*/

NETDEV\_EVENT\_RES\_TYPE\_DEVRIGHT = 10, /\* Device permission resource, corresponds to channel ID \*/

NETDEV\_EVENT\_RES\_TYPE\_ORG = 11, /\* Organization resource, corresponds to organization ID \*/

NETDEV\_EVENT\_RES\_TYPE\_ALARM\_TASK = 12, /\* Alarm task resource, corresponds to alarm task ID \*/

NETDEV\_EVENT\_RES\_TYPE\_SLAVE = 13, /\* Primary/replica resource (consistent with the server) \*/

NETDEV\_EVENT\_RES\_TYPE\_TVWALL = 14, /\* Video wall resource, corresponds to video wall ID \*/

NETDEV\_EVENT\_RES\_TYPE\_TVWALL\_SCENE = 15, /\* Video wall scene resource, corresponds to video wall ID \*/

NETDEV\_EVENT\_RES\_TYPE\_WND = 16, /\* Video wall window resource, corresponds to window ID \*/

NETDEV\_EVENT\_RES\_TYPE\_VIRTUAL\_LED = 17, /\* Video wall virtual LED resource, corresponds to virtual LED ID \*/

NETDEV\_EVENT\_RES\_TYPE\_BROADCAST\_CHANGE = 18, /\* Multicast group info change (value is consistent with the server) \*/

NETDEV\_EVENT\_RES\_TYPE\_LOGIC\_ORG = 19, /\* Virtual organization resource, corresponds to organization ID, used when deleting channels under a virtual organization \*/

NETDEV\_EVENT\_RES\_TYPE\_USER\_ROLE = 20, /\* User role resource, corresponds to user login handle \*/

NETDEV\_EVENT\_RES\_TYPE\_ROLE\_ORG = 21, /\* Role & organization tree resource, corresponds to user login handle \*/

NETDEV\_EVENT\_RES\_TYPE\_EMAP\_PIC = 22, /\* Image resource, corresponds to hot zone ID \*/

NETDEV\_EVENT\_RES\_TYPE\_PATROL = 23, /\* Patrol resource \*/

NETDEV\_EVENT\_RES\_TYPE\_RECORD = 24, /\* Recorded patrol resource \*/

NETDEV\_EVENT\_RES\_TYPE\_ACS\_PERSON = 25, /\* Access control person resource, corresponds to person ID \*/

NETDEV\_EVENT\_RES\_TYPE\_ACS\_PERSON\_CARD = 26, /\* Access control card resource, corresponds to person ID \*/

NETDEV\_EVENT\_RES\_TYPE\_TVWALL\_LIST = 27, /\* Video wall list, permission assigned to video wall \*/

NETDEV\_EVENT\_RES\_TYPE\_TVWALL\_SCENE\_SWITCH = 28, /\* Video wall scene switch \*/

NETDEV\_EVENT\_RES\_TYPE\_FACE\_LIB = 29, /\* Face library resource, corresponds to face library ID \*/

NETDEV\_EVENT\_RES\_TYPE\_FACE\_CUSTOM = 30, /\* Face library custom attribute, corresponds to attribute ID \*/

NETDEV\_EVENT\_RES\_TYPE\_FACE\_MEMBER = 31, /\* Face member resource, corresponds to face library ID \*/

NETDEV\_EVENT\_RES\_TYPE\_FACE\_GUARD = 32, /\* Face monitoring resource, corresponds to face monitoring ID \*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_DETECT = 33, /\* Smart detection resource, corresponds to smart detection type. Face recognition: 0 \*/

NETDEV\_EVENT\_RES\_TYPE\_MANUAL\_STATUS = 34, /\* Manual recording status \*/

NETDEV\_EVENT\_RES\_TYPE\_VEHICLE\_GUARD = 38, /\* Plate monitoring resource, corresponds to vehicle monitoring ID \*/

NETDEV\_EVENT\_RES\_TYPE\_CDN\_CHANNEL = 39, /\* CDN channel resource change. The changed info is not reported. The client queries the channel list proactively after receiving an event \*/

NETDEV\_EVENT\_RES\_TYPE\_FACE\_MEMBER\_SORT = 40, /\* Face member allocation resource, corresponds to face library ID \*/

NETDEV\_EVENT\_RES\_TYPE\_VEHICLE\_LIB = 41, /\* Vehicle library resource, corresponds to vehicle library ID \*/

NETDEV\_EVENT\_RES\_TYPE\_VEHICLE\_MEMBER\_SORT = 42, /\* Vehicle member allocation resource, corresponds to vehicle library ID \*/

NETDEV\_EVENT\_RES\_TYPE\_VEHICLE\_MEMBER = 43, /\* Vehicle member resource, corresponds to vehicle member ID \*/

NETDEV\_EVENT\_RES\_TYPE\_REMAIN\_PARK\_SPACE = 44, /\* Parking lot remaining parking space resource \*/

NETDEV\_EVENT\_RES\_TYPE\_ABNORMAL\_CHARGE = 45, /\* Abnormal charging rule resource \*/

NETDEV\_EVENT\_RES\_TYPE\_PARK\_RES\_CHANGE = 46, /\* Parking lot resource change \*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_COMMUNITY\_PERSON = 47, /\* Smart community person resource \*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_COMMUNITY\_CAR = 48, /\* Smart community vehicle resource \*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_COMMUNITY\_BUILDING = 49, /\* Smart community building resource \*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_COMMUNITY\_UNIT = 50, /\* Smart community unit resource \*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_COMMUNITY\_ROOM = 51, /\* Smart community room resource \*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_COMMUNITY\_ORG = 52, /\* Smart community resource organization (community) resource \*/

NETDEV\_EVENT\_RES\_TYPE\_TEMP\_VEHICLE\_PAY\_CHANGE = 53, /\* Temporary vehicle charging info change \*/

NETDEV\_EVENT\_RES\_TYPE\_THIRDPART\_PAYMENT = 54, /\* Parking lot third-party payment resource \*/

NETDEV\_EVENT\_RES\_TYPE\_PAYMENT\_WAIT = 55, /\* Payment in progress. Please wait \*/

NETDEV\_EVENT\_RES\_TYPE\_ENTRANCE\_LIST = 56, /\* Entrance & exit list. Permission to entrance & exit \*/

NETDEV\_EVENT\_RES\_TYPE\_VIEWPLAN\_RES = 57, /\* View plan. Corresponds to plan ID \*/

NETDEV\_EVENT\_RES\_TYPE\_SCENESPLAN\_RES = 58, /\* Scene plan. Corresponds to plan ID \*/

NETDEV\_EVENT\_RES\_TYPE\_ACS\_PERMISSION = 59, /\* Permission resource, used to change permission info \*/

NETDEV\_EVENT\_RES\_TYPE\_ACS\_GROUP = 60, /\* Access control group resource, used to change access control permission group (organization) \*/

NETDEV\_EVENT\_RES\_TYPE\_TVWALL\_AUDIO = 61, /\* Audio event \*/

NETDEV\_EVENT\_RES\_TYPE\_PAYMENT\_RRULE = 62, /\* Charging rule resource added manually \*/

NETDEV\_EVENT\_RES\_TYPE\_GIS = 63, /\* E-map resource \*/

NETDEV\_EVENT\_RES\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_EVENT\_RES\_TYPE\_E;

### People counting types

typedef enum tagNETDEVTrafficStatisticsType

{

NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_IN = 0, /\* Type: Enter \*/

NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_OUT = 1, /\* Type: Leave \*/

NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_ALL = 2, /\* Type: All \*/

NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_BUTT

}NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_E;

### People counting report types

typedef enum tagNETDEVFormType

{

NETDEV\_FORM\_TYPE\_DAY = 0, /\* Daily report \*/

NETDEV\_FORM\_TYPE\_WEEK = 1, /\* Weekly report \*/

NETDEV\_FORM\_TYPE\_MONTH = 2, /\* Monthly report \*/

NETDEV\_FORM\_TYPE\_YEAR = 3, /\* Yearly report \*/

NETDEV\_FORM\_TYPE\_BUTT

}NETDEV\_FORM\_TYPE\_E;

### People counting report types

typedef enum tagNETDEVTrafficStatisticsFormType

{

NETDEV\_TRAFFIC\_STAT\_FORM\_BY\_MINUTE = 0, /\* By minute \*/

NETDEV\_TRAFFIC\_STAT\_FORM\_BY\_HOUR = 1, /\* By hour \*/

NETDEV\_TRAFFIC\_STAT\_FORM\_BY\_DAY = 2, /\* By day \*/

NETDEV\_TRAFFIC\_STAT\_FORM\_BY\_MONTH = 3, /\* By month \*/

NETDEV\_TRAFFIC\_STAT\_FORM\_BY\_BUTT

}NETDEV\_TRAFFIC\_STATIC\_FORM\_TYPE\_E;

### Enumeration of resolutions

typedef enum tagNETDEVVideoFormat

{

NETDEV\_VIDEO\_FORMAT\_720P24 = 2, /\* 1280x720@24 \*/

NETDEV\_VIDEO\_FORMAT\_720P25 = 3, /\* 1280x720@25 \*/

NETDEV\_VIDEO\_FORMAT\_720P30 = 4, /\* 1280x720@30 \*/

NETDEV\_VIDEO\_FORMAT\_720P50 = 5, /\* 1280x720@50 \*/

NETDEV\_VIDEO\_FORMAT\_720P60 = 6, /\* 1280x720@60 \*/

NETDEV\_VIDEO\_FORMAT\_1080P24 = 10, /\* 1920x1080@24 \*/

NETDEV\_VIDEO\_FORMAT\_1080P25 = 11, /\* 1920x1080@25 \*/

NETDEV\_VIDEO\_FORMAT\_1080P30 = 12, /\* 1920x1080@30 \*/

NETDEV\_VIDEO\_FORMAT\_1080P50 = 13, /\* 1920x1080@50 \*/

NETDEV\_VIDEO\_FORMAT\_1080P60 = 14, /\* 1920x1080@60 \*/

NETDEV\_VIDEO\_FORMAT\_XGA60 = 15, /\* 1024x768@60 \*/

NETDEV\_VIDEO\_FORMAT\_SXGA60 = 16, /\* 1280x1024@60 \*/

NETDEV\_VIDEO\_FORMAT\_UXGA60 = 17, /\* 1600x1200@60 \*/

NETDEV\_VIDEO\_FORMAT\_SXGAP60HZ = 18,

NETDEV\_VIDEO\_FORMAT\_WXGAP60HZ = 19, /\* 1366x768@60 \*/

NETDEV\_VIDEO\_FORMAT\_WSXGA60HZ = 20, /\* 1440x900@60 \*/

NETDEV\_VIDEO\_FORMAT\_4K30 = 21, /\* 3840x2160@30 \*/

NETDEV\_VIDEO\_FORMAT\_4K60 = 22, /\* 3840x2160@60 \*/

NETDEV\_VIDEO\_FORMAT\_MUXGA60HZ = 23, /\* 1920x1200@60 \*/

NETDEV\_VIDEO\_FORMAT\_CUSTOM = 100, /\* Custom resolution \*/

NETDEV\_VIDEO\_FORMAT\_INVALID = 0xF

}NETDEV\_VIDEO\_FORMAT\_E;

### Enumeration of window splitting capability

typedef enum tagNETDEVPaneCap

{

NETDEV\_PANE\_UNSUPPORT = 0, /\* Not support \*/

NETDEV\_PANE\_SAME\_WND = 1, /\* Support splitting windows of the same size and position on one monitor \*/

NETDEV\_PANE\_ANY\_WND = 2, /\* Support splitting any window \*/

NETDEV\_PANE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_XW\_PANE\_CAP\_E;

### Enumeration of window roaming capability

typedef enum tagNETDEVRoamingCap

{

NETDEV\_ROAMING\_UNSUPPORT = 0, /\* Not support \*/

NETDEV\_ROAMING\_SINGLE\_WND = 1, /\* Supports roaming single split-window \*/

NETDEV\_ROAMING\_PANE\_WND = 2, /\* Supports roaming multiple split-windows \*/

NETDEV\_ROAMING\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_XW\_ROAMING\_CAP\_E;

### Enumeration of window opening capability

typedef enum tagNETDEVWndCap

{

NETDEV\_WINDOWS\_UNSUPPORT = 0, /\* Not support \*/

NETDEV\_WINDOWS\_SINGLE\_WND = 1, /\* Supports opening windows of the same size and position on one monitor \*/

NETDEV\_WINDOWS\_MULTI\_WND = 2, /\* Supports opening windows across multiple monitors, and windows expands to maximum window size \*/

NETDEV\_WINDOWS\_ANY\_WND = 3, /\* Supports opening windows randomly \*/

NETDEV\_WINDOWS\_INVALID =0xff /\* Invalid value \*/

}NETDEV\_XW\_WND\_CAP\_E;

### Enumeration of video output formats

typedef enum tagNETDEVXWLayoutNum

{

NETDEV\_XW\_LAYOUT\_TYPE\_\_SELF\_ADAPT = 0, /\* Adaptive \*/

NETDEV\_XW\_LAYOUT\_TYPE\_01 = 100, /\* 1 split window \*/

NETDEV\_XW\_LAYOUT\_TYPE\_02 = 200, /\* 2 split windows \*/

NETDEV\_XW\_LAYOUT\_TYPE\_03 = 300, /\* 3 split windows \*/

NETDEV\_XW\_LAYOUT\_TYPE\_03\_1 = 301, /\* 3 split windows, 2 at top, 1 at bottom \*/

NETDEV\_XW\_LAYOUT\_TYPE\_04 = 400, /\* 4 split windows \*/

NETDEV\_XW\_LAYOUT\_TYPE\_04\_1 = 401, /\* 4 split windows, 1+1+2 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_05 = 500, /\* 5 split windows, 1+3+1 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_05\_1 = 501, /\* 5 split windows, 1+1+3 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_05\_2 = 502, /\* 5 split windows, 1+3+1 (sequence different from 500) \*/

NETDEV\_XW\_LAYOUT\_TYPE\_05\_3 = 503, /\* 5 split windows, 1+1+3 (sequence different from 501) \*/

NETDEV\_XW\_LAYOUT\_TYPE\_05\_4 = 504, /\* 5 split windows, 3+1+1\*/

NETDEV\_XW\_LAYOUT\_TYPE\_05\_5 = 505, /\* 5 split windows, 1+4\*/

NETDEV\_XW\_LAYOUT\_TYPE\_06 = 600, /\* 6 split windows, 1+5 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_07 = 700, /\* 7 split windows, 1+6 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_07\_1 = 701, /\* 7 split windows, 3+1+3 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_07\_2 = 702, /\* 7 split windows, 3+1+3 (both 3: 2 at top, 1 at bottom) \*/

NETDEV\_XW\_LAYOUT\_TYPE\_08 = 800, /\* 8 split windows, 1+7 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_09 = 900, /\* 9 split windows \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10 = 1000, /\* 10 split windows, 1+9 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10\_1 = 1001, /\* 10 split windows, 2+8 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_13 = 1300, /\* 13 split windows, 1+12 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_13\_1 = 1301, /\* 13 split windows, 1+12 (sequence different from 1300) \*/

NETDEV\_XW\_LAYOUT\_TYPE\_16 = 1600, /\* 16 split windows \*/

NETDEV\_XW\_LAYOUT\_TYPE\_17 = 1700, /\* 17 split windows, 1+16 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_25 = 2500, /\* 25 split windows \*/

NETDEV\_XW\_LAYOUT\_TYPE\_32 = 3200, /\* 32 split windows, 3+1+28 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_36 = 3600, /\* 36 split windows \*/

NETDEV\_XW\_LAYOUT\_TYPE\_64 = 6400, /\* 64 split windows \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10001 = 10001, /\* Custom layout 1 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10002 = 10002, /\* Custom layout 2 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10003 = 10003, /\* Custom layout 3 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10004 = 10004, /\* Custom layout 4 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10005 = 10005, /\* Custom layout 5 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10006 = 10006, /\* Custom layout 6 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10007 = 10007, /\* Custom layout 7 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10008 = 10008, /\* Custom layout 8 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_INVALID = 0xffff

}NETDEV\_XW\_LAYOUT\_NUM\_E;

### Enumeration of video port types

typedef enum tagNETDEVVideoPortType

{

NETDEV\_VIDEO\_PORT\_BNC = 0, /\* BNC \*/

NETDEV\_VIDEO\_PORT\_VGA = 1, /\* VGA \*/

NETDEV\_VIDEO\_PORT\_HDMI = 2, /\* HDMI \*/

NETDEV\_VIDEO\_PORT\_DVI\_D = 3, /\* DVI\_D \*/

NETDEV\_VIDEO\_PORT\_DVI\_I = 4, /\* DVI\_D \*/

NETDEV\_VIDEO\_PORT\_DP = 5, /\* DVI\_D \*/

NETDEV\_VIDEO\_PORT\_SDI = 6, /\* DVI\_D \*/

NETDEV\_VIDEO\_PORT\_RJ45 = 7, /\* RJ45 \*/

NETDEV\_VIDEO\_PORT\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_VIDEO\_PORT\_TYPE\_E;

### Enumeration of audio channel ports

typedef enum tagNETDEVAudioPortType

{

NETDEV\_AUDIO\_PORT\_SINGLE = 0, /\* Single port \*/

NETDEV\_AUDIO\_PORT\_DOUBLE = 1, /\* Dual port \*/

NETDEV\_AUDIO\_PORT\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_AUDIO\_PORT\_TYPE\_E;

### Enumeration of serial ports

typedef enum tagNETDEVXWSerialType

{

NETDEV\_XW\_SERIAL\_TYPE\_RS232 = 0, /\* 232 \*/

NETDEV\_XW\_SERIAL\_TYPE\_RS422 = 1, /\* 422 \*/

NETDEV\_XW\_SERIAL\_TYPE\_RS485 = 2, /\* 485 \*/

NETDEV\_XW\_SERIAL\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_XW\_SERIAL\_TYPE\_E;

### Enumeration of virtual LED types

typedef enum tagNETDEVLEDType

{

NETDEV\_LED\_TYPE\_STRING = 0 /\* Character \*/

}NETDEV\_XW\_LED\_TYPE\_E;

### Enumeration of optional command words for modifying virtual LEDs

typedef enum tagNETDEVLedCmd

{

NETDEV\_LED\_ENABLE = 0x0001, /\* Window layout udwEnable \*/

NETDEV\_LED\_TYPE = 0x0002, /\* Virtual LED type dwType \*/

NETDEV\_LED\_AREA = 0x0004, /\* Virtual LED position info stArea \*/

NETDEV\_LED\_BACKGROUND = 0x0008, /\* Background color stBackground \*/

NETDEV\_LED\_FONTINFO = 0x0010, /\* Text description stFontInfo \*/

NETDEV\_LED\_TEXT = 0x0020, /\* Text content udwTestLineNum szText\*/

NETDEV\_LED\_ALL = 0x00FF /\* All all\*/

} NETDEV\_XW\_LED\_CMD\_E;

### Enumeration of virtual LED font format

typedef enum tagNETDEVLEDFontFormat

{

NETDEV\_LED\_FONT\_SONG = 0, /\* SimSun \*/

NETDEV\_LED\_FONT\_FORMAT\_SIMHEI = 1, /\* SimHei \*/

NETDEV\_LED\_FONT\_FORMAT\_KAITI = 2, /\* KaiTi \*/

NETDEV\_LED\_FONT\_FORMAT\_Arial = 3, /\* Arial \*/

NETDEV\_LED\_FONT\_FORMAT\_FANGSONG = 4, /\* FangSong \*/

NETDEV\_LED\_FONT\_FORMAT\_LISHU = 5, /\* LiSu \*/

NETDEV\_LED\_FONT\_FORMAT\_YOUYUAN = 6 /\* YouYuan \*/

}NETDEV\_XW\_LED\_FONT\_FORMAT\_E;

### Enumeration of virtual LED font sizes

typedef enum tagNETDEVXWLEDFontSize

{

NETDEV\_LED\_FONT\_SIZE\_48X48 = 0, /\* 48X48 \*/

NETDEV\_LED\_FONT\_SIZE\_64X64 = 1, /\* 64X64 \*/

NETDEV\_LED\_FONT\_SIZE\_96X96 = 2, /\* 96X96 \*/

NETDEV\_LED\_FONT\_SIZE\_128X128 = 3, /\* 128X128 \*/

NETDEV\_LED\_FONT\_SIZE\_192X192 = 4, /\* 192X192 \*/

NETDEV\_LED\_FONT\_SIZE\_256X256 = 5, /\* 256X256 \*/

NETDEV\_LED\_FONT\_SIZE\_512X512 = 6, /\* 512X512 \*/

NETDEV\_FONT\_SIZE\_SELF\_ADAPTION = 7 /\* Adaptive \*/

}NETDEV\_XW\_LED\_FONT\_SIZE\_E;

### Enumeration of virtual LED font spacing

typedef enum tagNETDEVXWLEDFontGap

{

NETDEV\_LED\_FONT\_GAP\_NORMAL = 0, /\* Normal \*/

NETDEV\_LED\_FONT\_GAP\_ONE = 1, /\* 2/4 height \*/

NETDEV\_LED\_FONT\_GAP\_TWO = 2, /\* 4/4 height \*/

NETDEV\_LED\_FONT\_GAP\_THREE = 3, /\* 6/4 height \*/

NETDEV\_LED\_FONT\_GAP\_FOUR = 4, /\* 8/4 height \*/

NETDEV\_LED\_FONT\_GAP\_FIVE = 5, /\* 1/4 height \*/

NETDEV\_LED\_FONT\_GAP\_SIX = 6, /\* 3/4 height \*/

NETDEV\_LED\_FONT\_GAP\_SEVEN = 7, /\* 5/4 height \*/

NETDEV\_LED\_FONT\_GAP\_EIGHT = 8, /\* 7/4 height \*/

NETDEV\_LED\_FONT\_GAP\_NINE = 9, /\* 10/4 height \*/

NETDEV\_LED\_FONT\_GAP\_INVALID = 0xFF /\* Invalid value (Adaptive) \*/

}NETDEV\_XW\_LED\_FONT\_GAP\_E;

### Enumeration of horizontal text alignment modes

typedef enum tagNETDEVHorizontalAlignMode

{

NETDEV\_HORIZONTAL\_ALIGN\_LEFT = 0, /\* Left alignment \*/

NETDEV\_HORIZONTAL\_ALIGN\_MIDDLE = 1, /\* Center alignment \*/

NETDEV\_HORIZONTAL\_ALIGN\_RIGHT = 2 /\* Right alignment \*/

}NETDEV\_XW\_HORIZONTAL\_ALIGN\_MODE\_E;

### Enumeration of vertical text alignment modes

typedef enum tagNETDEVVerticalAlignMode

{

NETDEV\_VERTICAL\_ALIGN\_TOP = 0, /\* Align top \*/

NETDEV\_VERTICAL\_ALIGN\_MIDDLE = 1, /\* Align middle \*/

NETDEV\_VERTICAL\_ALIGN\_BOTTOM = 2 /\* Align bottom \*/

}NETDEV\_XW\_VERTICAL\_ALIGN\_MODE\_E;

### Enumeration of text scroll modes

typedef enum tagNETDEVLEDScrollingMode

{

NETDEV\_LED\_SCROLL\_STATIC = 0, /\* Static \*/

NETDEV\_LED\_SCROLL\_RIGHT\_TO\_LEFT = 1, /\* Right to left \*/

NETDEV\_LED\_SCROLL\_LEFT\_TO\_RIGHT = 2, /\* Left to right \*/

NETDEV\_LED\_SCROLL\_BOTTOM\_TO\_TOP = 3, /\* Bottom to top \*/

NETDEV\_LED\_SCROLL\_TOP\_TO\_BOTTOM = 4 /\* Top to bottom \*/

}NETDEV\_XW\_LED\_SCROLLING\_MODE\_E;

### Enumeration of window magnification types

typedef enum tagNETDEVZoomType

{

NETDEV\_XW\_ZOOM\_TYPE\_NO = 0x00, /\* No magnification \*/

NETDEV\_XW\_ZOOM\_TYPE\_SPLIT = 0x01, /\* Magnify split window \*/

NETDEV\_XW\_ZOOM\_TYPE\_WND = 0x02, /\* Magnify window \*/

NETDEV\_XW\_ZOOM\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_XW\_ZOOM\_TYPE\_E;

### Enumeration of window types

typedef enum tagNETDEVTVWallWndType

{

NETDEV\_WINDOWS\_TYPE\_NORMAL = 0, /\* Normal window \*/

NETDEV\_WINDOWS\_TYPE\_ALARM = 1, /\* Alarm window \*/

NETDEV\_WINDOWS\_TYPE\_INVALID = 0xFF /\* Invalid status \*/

}NETDEV\_TVWALL\_WND\_TPYE\_E;

### Enumeration of zoom statuses

typedef enum tagNETDEVScaleEnabled

{

NETDEV\_SCALE\_ENABLED\_NO = 0, /\* No scaling \*/

NETDEV\_SCALE\_ENABLED\_IN = 1, /\* Scale \*/

NETDEV\_SCALE\_ENABLED\_FULL = 2, /\* Stretch \*/

NETDEV\_SCALE\_ENABLED\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_SCALE\_ENABLED\_E;

### Enumeration of video service in split window

typedef enum tagNETDEVSessionType

{

NETDEV\_SESSION\_TYPE\_FREE = 0, /\* Idle \*/

NETDEV\_SESSION\_TYPE\_LIVE\_ON\_WALL = 1, /\* Live view on video wall \*/

NETDEV\_SESSION\_TYPE\_PLAYBACK\_ON\_WALL = 2, /\* Playback on video wall \*/

NETDEV\_SESSION\_TYPE\_TURNING\_ON\_WALL = 3 /\* Sequence on video wall \*/

}NETDEV\_SESSION\_TYPE\_E;

### Enumeration of screen magnification types

typedef enum tagNETDEVXWScreenZoomdType

{

NETDEV\_XW\_SCREEN\_ZOOM\_TYPE\_PANE = 0, /\* Magnify split-window \*/

NETDEV\_XW\_SCREEN\_ZOOM\_TYPE\_WIN = 1, /\*Magnify window \*/

NETDEV\_XW\_SCREEN\_ZOOM\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_XW\_SCREEN\_ZOOM\_TYPE\_E;

### Enumeration of screen magnification modes

typedef enum tagNETDEVXWScreenZoomdMode

{

NETDEV\_XW\_SCREEN\_ZOOM\_MODE\_OFF = 0, /\* Restore \*/

NETDEV\_XW\_SCREEN\_ZOOM\_MODE\_ON = 1, /\* Magnify \*/

NETDEV\_XW\_SCREEN\_ZOOM\_MODE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_XW\_SCREEN\_ZOOM\_MODE\_E;

### Enumeration of media transport protocols

typedef enum tagNETDEVLapiTransProtocal

{

NETDEV\_LAPI\_TRANS\_PROTOCAL\_TCP = 0, /\* TCP \*/

NETDEV\_LAPI\_TRANS\_PROTOCAL\_UDP = 1 /\* UDP \*/

}NETDEV\_LAPI\_TRANS\_PROTOCAL\_E;

### Enumeration of live view types

typedef enum tagNETDEVRealPlayType

{

NETDEV\_TYPE\_NORMAL = 0, /\* Default: normal live view \*/

NETDEV\_TYPE\_ALARM = 1, /\* Alarm triggered live view \*/

NETDEV\_TYPE\_LOCAL = 2 /\* Local live view \*/

}NETDEV\_XW\_REALPLAY\_TYPE\_E;

### Enumeration of decoding types

typedef enum tagNETDEVDecodeType

{

NETDEV\_DECODE\_TYPE\_CLIENT = 0, /\* Decoder works as the client, active decoding \*/

NETDEV\_DECODE\_TYPE\_SERVER = 1 /\* Decoder works as the server, passive decoding \*/

}NETDEV\_XW\_DECODE\_TYPE\_E;

### Enumeration of encryption types

typedef enum tagNetDevAuthModel

{

NETDEV\_AUTH\_MODEL\_RSA\_10 = 0, /\* RSA, decimal \*/

NETDEV\_AUTH\_MODEL\_PRI = 1, /\* Common encryption \*/

NETDEV\_AUTH\_MODEL\_RSA\_16 = 2 /\* RSA, hexadecimal \*/

}NETDEV\_AUTH\_MODEL\_E;

### Enumeration of text positions

typedef enum tagNETDEVContentPosition

{

NETDEV\_XW\_CONTENT\_POSITION\_TOP = 0, /\* Top \*/

NETDEV\_XW\_CONTENT\_POSITION\_MIDDLE = 1, /\* Middle \*/

NETDEV\_XW\_CONTENT\_POSITION\_BOTTOM = 2 /\* Bottom \*/

}NETDEV\_XW\_CONTENT\_POSITION\_E;

### Enumeration of resource management types

typedef enum tagNETDEVXWManageType

{

NETDEV\_XW\_MANAGE\_TYPE\_OTHER = 0, /\* Other \*/

NETDEV\_XW\_MANAGE\_TYPE\_XWARE = 1, /\* Xware device manages resource \*/

NETDEV\_XW\_MANAGE\_TYPE\_INVALID = 0XFF /\* Invalid value \*/

}NETDEV\_XW\_MANAGE\_TYPE\_E;

### Enumeration of stream types

typedef enum tagNETDEVStreamType

{

NETDEV\_STREAM\_TYPE\_RTP\_UNICAST = 0, /\* Unicast \*/

NETDEV\_STREAM\_TYPE\_RTP\_MULTICAST = 1, /\* Multicast \*/

NETDEV\_STREAM\_TYPE\_RTP\_INVALID = 0xFF

}NETDEV\_STREAM\_TYPE\_E;

### Enumeration of scene modification types

typedef enum tagNETDEVXWSceneModifyType

{

NETDEV\_XW\_SCENE\_MODIFY\_TYPE\_NAME = 0, /\* Change the scene name \*/

NETDEV\_XW\_SCENE\_MODIFY\_TYPE\_SAVE\_AS = 1, /\* Save the scene as another scene \*/

NETDEV\_XW\_SCENE\_MODIFY\_TYPE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_XW\_SCENE\_MODIFY\_TYPE\_E;

### Enumeration of scene modes

typedef enum tagNETDEVSceneSessionMode

{

NETDEV\_SCENE\_SESSION\_MODE\_CLASSICS = 0, /\* Classic mode \*/

NETDEV\_SCENE\_SESSION\_MODE\_ADAPTIVE\_IN\_SIGNAL\_SRC = 1, /\* Adaptive input signal source mode \*/

NETDEV\_SCENE\_SESSION\_MODE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_SCENE\_SESSION\_MODE\_E;

### Enumeration of background image types

typedef enum tagNETDEVBaseType

{

NETDEV\_BASE\_TYPE\_JPEG = 1 /\* JPEG \*/

}NETDEV\_XW\_BASE\_TYPE\_E;

### Enumeration of sequence statuses

typedef enum tagNETDEVXWSequenceResStatus

{

NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_RUNNING = 0, /\* Running \*/

NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_PAUSE = 1, /\* Paused \*/

NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_STOP = 2, /\* Stopped \*/

NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_E;

### Enumeration of sequence playing control

typedef enum tagNETDEVXWSequencePlayControlCmd

{

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_STOP = 0, /\* Stop \*/

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_START = 1, /\* Start \*/

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_PAUSE = 2, /\* Pause \*/

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_RESUME = 3, /\* Resume \*/

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_PAGEUP = 4, /\* Previous screen \*/

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_PAGEDOWN = 5, /\* Next screen \*/

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_E;

### Enumeration of scene sequence plan modes

typedef enum tagNETDEVXWScenesPlanMode

{

NETDEV\_XW\_SCENES\_PLAN\_MODE\_TIMER = 0, /\* Switch according to a timer \*/

NETDEV\_XW\_SCENES\_PLAN\_MODE\_TIME\_TABLE = 1, /\* Switch according to a 24H schedule \*/

NETDEV\_XW\_SCENES\_PLAN\_MODE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_XW\_SCENES\_PLAN\_MODE\_E;

### Enumeration of encoding formats

typedef enum tagNETDEVVideoEncoding

{

NETDEV\_VIDEOENCODING\_H264 = 0,

NETDEV\_VIDEOENCODING\_H265 = 1,

NETDEV\_VIDEOENCODING\_MPEG4 = 2,

NETDEV\_VIDEOENCODING\_MPEG2 = 3,

NETDEV\_VIDEOENCODING\_MJPEG = 4

}NETDEV\_XW\_VIDEOENCODING\_E;

### Enumeration of command words for modifying scene windows

typedef enum tagNETDEVWndCommond

{

NETDEV\_WND\_PANE\_MOD = 0x0001, /\* Layout udwPaneMod \*/

NETDEV\_WND\_LAYER = 0x0002, /\* Layer udwLayer \*/

NETDEV\_WND\_TRANSPARENCY = 0x0004, /\* Transparency udwTransparency \*/

NETDEV\_WND\_AREA = 0x0008, /\* Location info stArea \*/

NETDEV\_WND\_NAME = 0x0010, /\* Window name name \*/

NETDEV\_WND\_STATUS = 0x0020, /\* Lock or not \*/

NETDEV\_WND\_WND\_TYPE = 0x0040, /\* Window type \*/

NETDEV\_WND\_DC\_CHI\_ID = 0x0080, /\* DC channel ID \*/

NETDEV\_WND\_ZOOM\_TYPE = 0x0100, /\* Magnification type \*/

NETDEV\_WND\_SPLIT\_INDEX = 0x0200, /\* Split-window ID \*/

NETDEV\_WND\_SCHEME\_RES\_ID = 0x0400, /\* Sequence resource \*/

NETDEV\_WND\_SEQ\_STATUS = 0x0800, /\* Sequence status \*/

NETDEV\_WND\_PANE\_INFO\_NUM = 0x1000, /\* Number of split windows \*/

NETDEV\_WND\_SCALE\_STATUS = 0x2000, /\* Scale status \*/

NETDEV\_WND\_ALL = 0xFFFF /\* All configuration all \*/

} NETDEV\_XW\_WND\_CMD\_E;

### Enumeration of sequence statuses

typedef enum tagNETDEVSeqStatus

{

NETDEV\_SEQ\_STATUS\_EMPTY = 0, /\* No sequence \*/

NETDEV\_SEQ\_STATUS\_RUNNING = 1, /\* Sequencing \*/

NETDEV\_SEQ\_STATUS\_STOP = 2, /\* Sequence stopped \*/

NETDEV\_SEQ\_STATUS\_INVALID = 0xFF /\* Invalid status \*/

}NETDEV\_SEQ\_STATUS\_E;

### Enumeration of stream service types

typedef enum tagNETDEVVideoType

{

NETDEV\_PLAYBACK\_REMOTE = 0, /\* Remote playback \*/

NETDEV\_REALPLAY\_REMOTE = 1, /\* Remote live view \*/

NETDEV\_REALPLAY\_LOCAL = 2, /\* Local live view \*/

NETDEV\_SEQUENCE\_PANE = 3, /\* Sequence in one split-window currently unavailable \*/

NETDEV\_SEQUENCE\_WND = 4, /\* Sequence in window \*/

NETDEV\_REALPLAY\_ALARM = 5 /\* Alarm-triggered live view \*/

}NETDEV\_XW\_VIDEO\_TYPE\_E;

### Enumeration of access control person management commands

typedef enum tagNETDEVACSPersonCommondType

{

NETDEV\_ACS\_PERSON\_COMMOND\_TYPE\_GET = 0, /\* Get \*/

NETDEV\_ACS\_PERSON\_COMMOND\_TYPE\_MOD = 1, /\* Modify \*/

NETDEV\_ACS\_PERSON\_COMMOND\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_ACS\_PERSON\_COMMOND\_TYPE\_E;

### Enumeration of alarm point channel control commands

typedef enum tagNETDEVDoorCtrlActionType

{

NETDEV\_DOORCTRL\_ACTION\_TYPE\_OPEN = 0, /\* Open door \*/

NETDEV\_DOORCTRL\_ACTION\_TYPE\_CLOSE = 1, /\* Close door \*/

NETDEV\_DOORCTRL\_ACTION\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_DOORCTRL\_ACTION\_TYPE\_E;

### Enumeration of manual linkage action types

typedef enum tagNETDEVlinkageType

{

NETDEV\_LINKAGE\_TYPE\_CLICK\_TRACK = 0, /\* Track \*/

NETDEV\_LINKAGE\_TYPE\_ENLARGE\_FRAME = 1, /\* Drag to zoom \*/

NETDEV\_LINKAGE\_TYPE\_3D\_LOCATE = 2, /\* 3D positioning \*/

NETDEV\_LINKAGE\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_LINKAGE\_TYPE\_E;

### Enumeration of face member ID types

typedef enum tagNETDEVFaceMemberIDType

{

NETDEV\_FACE\_MEMBER\_ID\_TYPE\_ID\_CARD = 0, /\* ID card \*/

NETDEV\_FACE\_MEMBER\_ID\_TYPE\_IC\_CARD = 1, /\* IC card \*/

NETDEV\_FACE\_MEMBER\_ID\_TYPE\_PASSPORT = 2, /\* Passport \*/

NETDEV\_FACE\_MEMBER\_ID\_TYPE\_DRIVING = 3, /\* Driver’s license \*/

NETDEV\_FACE\_MEMBER\_ID\_TYPE\_OTHER = 99, /\* Other \*/

NETDEV\_FACE\_MEMBER\_ID\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_FACE\_MEMBER\_ID\_TYPE\_E;

### Enumeration of image types

typedef enum tagNETDEVTMSPersionImageType

{

NETDEV\_TMS\_PERSION\_IMAGE\_TYPE\_FULL\_VIEW = 1, /\* Complete image \*/

NETDEV\_TMS\_PERSION\_IMAGE\_TYPE\_FACE = 2, /\* Face image \*/

NETDEV\_TMS\_PERSION\_IMAGE\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_TMS\_PERSION\_IMAGE\_TYPE\_E;

### Enumeration of image formats

typedef enum tagNETDEVTMSPersionImageFormatType

{

NETDEV\_TMS\_PERSION\_IMAGE\_FORMAT\_JPG = 1, /\* JPEG \*/

NETDEV\_TMS\_PERSION\_IMAGE\_FORMAT\_BMP = 2, /\* BMP \*/

NETDEV\_TMS\_PERSION\_IMAGE\_FORMAT\_PNG = 3, /\* PNG \*/

NETDEV\_TMS\_PERSION\_IMAGE\_FORMAT\_GIF = 4, /\* GIF \*/

NETDEV\_TMS\_PERSION\_IMAGE\_FORMAT\_TIFF = 5, /\* TIFF \*/

NETDEV\_TMS\_PERSION\_IMAGE\_FORMAT\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_TMS\_PERSION\_IMAGE\_FORMAT\_E;

### Enumeration of face pass-through record types

typedef enum tagNETDEVFacePassRecordType

{

NETDEV\_TYPE\_FACE\_PASS\_SNAPSHOT = 0, /\* Face snapshot \*/

NETDEV\_TYPE\_FACE\_PASS\_COM\_SUCCESS = 1, /\* Comparison success alarm \*/

NETDEV\_TYPE\_FACE\_PASS\_COM\_FAIL = 2, /\* Comparison failure alarm \*/

NETDEV\_TYPE\_FACE\_PASS\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_FACE\_PASS\_RECORD\_TYPE\_E;

### Enumeration of age ranges

typedef enum tagNETDEVAgeRange

{

NETDEV\_AGE\_RANGE\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_AGE\_RANGE\_CHILD = 1, /\* Child \*/

NETDEV\_AGE\_RANGE\_JUVENILE = 2, /\* Juvenile \*/

NETDEV\_AGE\_RANGE\_Youth = 3, /\* Youth \*/

NETDEV\_AGE\_RANGE\_MIDDLEAGE = 4, /\* Middle age \*/

NETDEV\_AGE\_RANGE\_OLDAGE = 5, /\* Senior \*/

NETDEV\_AGE\_RANGE\_INVALID = 0xFF /\* Invalid \*/

}NETDEV\_AGE\_RANGE\_E;

### Enumeration of glasses wearing status

typedef enum tagNETDEVGlassFlag

{

NETDEV\_GLASS\_FLAG\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_GLASS\_FLAG\_NO = 1, /\* No glasses\*/

NETDEV\_GLASS\_FLAG\_YES = 2, /\* With glasses \*/

NETDEV\_GLASS\_FLAG\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_GLASS\_FLAG\_E;

### Enumeration of glasses styles

typedef enum tagNETDEVGlassesStyle

{

NETDEV\_GLASSES\_STYLE\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_GLASSES\_STYLE\_GENERAL = 1, /\* General glasses \*/

NETDEV\_GLASSES\_STYLE\_SUNLIGHT = 2, /\* Sunglasses \*/

NETDEV\_GLASSES\_STYLE\_OTHER = 99, /\* Other \*/

NETDEV\_GLASSES\_STYLE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_GLASSES\_STYLE\_E;

### Enumeration of sleeve lengths

typedef enum tagNETDEVSleevesLength

{

NETDEV\_SLEEVES\_LENGTH\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_SLEEVES\_LENGTH\_SHORT = 1, /\* Short sleeves \*/

NETDEV\_SLEEVES\_LENGTH\_LONG = 2, /\* Long sleeves \*/

NETDEV\_SLEEVES\_LENGTH\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_SLEEVES\_LENGTH\_E;

### Enumeration of clothes colors

typedef enum tagNETDEVClothesColor

{

NETDEV\_CLOTHES\_COLOR\_BLACK\_E = 0, /\* Black \*/

NETDEV\_CLOTHES\_COLOR\_WHITE\_E = 1, /\* White \*/

NETDEV\_CLOTHES\_COLOR\_GRAY\_E = 2, /\* Grey \*/

NETDEV\_CLOTHES\_COLOR\_RED\_E = 3, /\* Red \*/

NETDEV\_CLOTHES\_COLOR\_BLUE\_E = 4, /\* Blue \*/

NETDEV\_CLOTHES\_COLOR\_YELLOW\_E = 5, /\* Yellow \*/

NETDEV\_CLOTHES\_COLOR\_ORANGE\_E = 6, /\* Orange \*/

NETDEV\_CLOTHES\_COLOR\_BROWN\_E = 7, /\* Brown \*/

NETDEV\_CLOTHES\_COLOR\_GREEN\_E = 8, /\* Green \*/

NETDEV\_CLOTHES\_COLOR\_PURPLE\_E = 9, /\* Purple \*/

NETDEV\_CLOTHES\_COLOR\_CYAN\_E = 10, /\* Cyan \*/

NETDEV\_CLOTHES\_COLOR\_PINK\_E = 11, /\* Pink \*/

NETDEV\_CLOTHES\_COLOR\_TRANSPARENT\_E = 12, /\* Transparent \*/

NETDEV\_CLOTHES\_COLOR\_SILVERYWHITE\_E = 13, /\* Silver white \*/

NETDEV\_CLOTHES\_COLOR\_DARK\_E = 14, /\* Dark color \*/

NETDEV\_CLOTHES\_COLOR\_LIGHT\_E = 15, /\* Light color \*/

NETDEV\_CLOTHES\_COLOR\_COLOURLESS = 16, /\* Colorless\*/

NETDEV\_CLOTHES\_COLOR\_YELLOWGREEN = 17, /\* Yellow&Green \*/

NETDEV\_CLOTHES\_COLOR\_GRADUALGREEN = 18, /\* Gradual green (or gradient green) \*/

NETDEV\_CLOTHES\_COLOR\_OTHER\_E = 99, /\* Other \*/

NETDEV\_CLOTHES\_COLOR\_UNKNOW\_E = 100, /\* Unknown \*/

NETDEV\_CLOTHES\_COLOR\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_CLOTHES\_COLOR\_E;

### Enumeration of trousers length

typedef enum tagNETDEVTrousersLength

{

NETDEV\_TROUSERS\_LENGTH\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_TROUSERS\_LENGTH\_SHORT = 1, /\* Short \*/

NETDEV\_TROUSERS\_LENGTH\_LONG = 2, /\* Long \*/

NETDEV\_TROUSERS\_LENGTH\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_TROUSERS\_LENGTH\_E;

### Enumeration of captured body directions

typedef enum tagNETDEVBodyToward

{

NETDEV\_BODY\_TOWARD\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_BODY\_TOWARD\_POSITIVE = 1, /\* Positive \*/

NETDEV\_BODY\_TOWARD\_SIDE = 2, /\* Side \*/

NETDEV\_BODY\_TOWARD\_BACK = 3, /\* Back \*/

NETDEV\_BODY\_TOWARD\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_BODY\_TOWARD\_E;

### Enumeration of shoe tube lengths

typedef enum tagNETDEVShoesTubeLength

{

NETDEV\_SHOES\_TUBE\_LENGTH\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_SHOES\_TUBE\_LENGTH\_LONG = 1, /\* Long tubes \*/

NETDEV\_SHOES\_TUBE\_LENGTH\_SHORT = 2, /\* Short tubes/ordinary shoes \*/

NETDEV\_SHOES\_TUBE\_LENGTH\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_SHOES\_TUBE\_LENGTH\_E;

### Enumeration of hair lengths

typedef enum tagNETDEVHairLength

{

NETDEV\_HAIR\_LENGTH\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_HAIR\_LENGTH\_LONG = 1, /\* Long hair \*/

NETDEV\_HAIR\_LENGTH\_SHORT = 2, /\* Short hair \*/

NETDEV\_HAIR\_LENGTH\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HAIR\_LENGTH\_E;

### Enumeration of bag carrying status

typedef enum tagNETDEVBagFlag

{

NETDEV\_BAG\_FLAG\_NO = 0, /\* No bag \*/

NETDEV\_BAG\_FLAG\_CARRY = 1, /\* Carrying bag \*/

NETDEV\_BAG\_FLAG\_BACK = 2, /\* Backpack \*/

NETDEV\_BAG\_FLAG\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_BAG\_FLAG\_E;

### Enumeration of mask wearing statuses

typedef enum tagNETDEVPersonMaskFlag

{

NETDEV\_PERSON\_MASK\_FLAG\_NOT\_WEAR = 1, /\* Not wearing mask \*/

NETDEV\_PERSON\_MASK\_FLAG\_WEAR = 2, /\* Wearing mask \*/

NETDEV\_PERSON\_MASK\_FLAG\_UNKNOW = 255, /\* Unknown \*/

NETDEV\_PERSON\_MASK\_FLAG\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_PERSON\_MASK\_FLAG\_E;

### Enumeration of coat textures

typedef enum tagNETDEVCoatTexture

{

NETDEV\_CLOTHES\_TEXTURE\_NO\_PATTERNS = 1, /\* No pattern \*/

NETDEV\_CLOTHES\_TEXTURE\_EXIST\_PATTERNS = 2, /\* With pattern \*/

NETDEV\_CLOTHES\_TEXTURE\_UNKNOW = 255, /\* Unknown \*/

NETDEV\_CLOTHES\_TEXTURE\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_CLOTHES\_TEXTURE\_E;

### Enumeration of person movement directions

typedef enum tagNETDEVMoveDirection

{

NETDEV\_MOVE\_DIRECTION\_STATIC = 1, /\* Static \*/

NETDEV\_MOVE\_DIRECTION\_UP = 2, /\* Up \*/

NETDEV\_MOVE\_DIRECTION\_DOWN = 3, /\* Down \*/

NETDEV\_MOVE\_DIRECTION\_LEFT = 4, /\* Left \*/

NETDEV\_MOVE\_DIRECTION\_RIGHT = 5, /\* Right \*/

NETDEV\_MOVE\_DIRECTION\_LEFTUP = 6, /\* Up left \*/

NETDEV\_MOVE\_DIRECTION\_LEFTDOWN = 7, /\* Down left \*/

NETDEV\_MOVE\_DIRECTION\_RIGHTUP = 8, /\* Up right \*/

NETDEV\_MOVE\_DIRECTION\_RIGHTDOWN = 9, /\* Down right \*/

NETDEV\_MOVE\_DIRECTION\_UNKNOW = 255, /\* Unknown \*/

NETDEV\_MOVE\_DIRECTION\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_MOVE\_DIRECTION\_E;

### Enumeration of speed types of non-motor vehicles in structured scenes

typedef enum tagNETDEVSpeedType

{

NETDEV\_SPEED\_TYPE\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_SPEED\_TYPE\_STATIC = 1, /\* Static \*/

NETDEV\_SPEED\_TYPE\_SLOW = 2, /\* Slow \*/

NETDEV\_SPEED\_TYPE\_MEDIUM = 3, /\* Medium \*/

NETDEV\_SPEED\_TYPE\_FAST = 4, /\* Fast \*/

NETDEV\_SPEED\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_SPEED\_TYPE\_E;

### Enumeration of movement directions (relative to image) of non-motor vehicles in structured scenes

typedef enum tagNETDEVImageDirection

{

NETDEV\_IMAGE\_DIRECTION\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_IMAGE\_DIRECTION\_STATIC = 1, /\* Static \*/

NETDEV\_IMAGE\_DIRECTION\_UP = 2, /\* Up \*/

NETDEV\_IMAGE\_DIRECTION\_DOWN = 3, /\* Down \*/

NETDEV\_IMAGE\_DIRECTION\_LEFT = 4, /\* Left \*/

NETDEV\_IMAGE\_DIRECTION\_RIGH = 5, /\* Right \*/

NETDEV\_IMAGE\_DIRECTION\_LEFTUP = 6, /\* Up left \*/

NETDEV\_IMAGE\_DIRECTION\_LEFTDOWN = 7, /\* Down left \*/

NETDEV\_IMAGE\_DIRECTION\_RIGHTUP = 8, /\* Up right \*/

NETDEV\_IMAGE\_DIRECTION\_RIGHTDOWN = 9, /\* Down right \*/

NETDEV\_IMAGE\_DIRECTION\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_IMAGE\_DIRECTION\_E;

### Enumeration of non-motor vehicle types

typedef enum tagNETDEVNonVehType

{

NETDEV\_NON\_VEH\_TYPE\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_NON\_VEH\_TYPE\_BICYCLE = 1, /\* Bicycle \*/

NETDEV\_NON\_VEH\_TYPE\_TRIYCLE = 2, /\* Tricycle \*/

NETDEV\_NON\_VEH\_TYPE\_MOTORCYCLE = 3, /\* Motorcycle \*/

NETDEV\_NON\_VEH\_TYPE\_ELECTRIC\_BICYCLE = 4, /\* Electric bicycle \*/

NETDEV\_NON\_VEH\_TYPE\_TWOWHEEL\_VEHICLE = 5, /\* Two-wheel (motorcycle/bicycle/electric bicycle) \*/

NETDEV\_NON\_VEH\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_NON\_VEH\_TYPE\_E;

### Enumeration of vehicle types

typedef enum tagNETDEVVehicleType

{

NETDEV\_VEHICLE\_TYPE\_TRICYCLE\_E = 0, /\* 3-wheel vehicle \*/

NETDEV\_VEHICLE\_TYPE\_MOTOR\_BUS\_E = 1, /\* Bus \*/

NETDEV\_VEHICLE\_TYPE\_MIDDLE\_E = 2, /\* Mid-size vehicle \*/

NETDEV\_VEHICLE\_TYPE\_SMALL\_E = 3, /\* Small vehicle \*/

NETDEV\_VEHICLE\_TYPE\_BIG\_E = 4, /\* Large vehicle \*/

NETDEV\_VEHICLE\_TYPE\_TWOWHEELVEH = 5, /\* 2-wheel vehicle \*/

NETDEV\_VEHICLE\_TYPE\_MOTORCYCLE\_E = 6, /\* Motorcycle \*/

NETDEV\_VEHICLE\_TYPE\_TRACTOR\_E = 7, /\* Tractor \*/

NETDEV\_VEHICLE\_TYPE\_AGRICULTURAL\_E = 8, /\* Farm truck \*/

NETDEV\_VEHICLE\_TYPE\_SEADAN = 9, /\* Sedan \*/

NETDEV\_VEHICLE\_TYPE\_SUV\_E = 10, /\* SUV \*/

NETDEV\_VEHICLE\_TYPE\_VAN\_E = 11, /\* Minibus \*/

NETDEV\_VEHICLE\_TYPE\_SMALLTRUCK\_E = 12, /\* Van \*/

NETDEV\_VEHICLE\_TYPE\_MEDIUMCAR\_E = 13, /\* Mid-sized bus/medium bus \*/

NETDEV\_VEHICLE\_TYPE\_LARGEBUS\_E = 14, /\* Large bus \*/

NETDEV\_VEHICLE\_TYPE\_LARGETRUCK\_E = 15, /\* Large truck \*/

NETDEV\_VEHICLE\_TYPE\_PICKUPTRUCK\_E = 16, /\* Pickup \*/

NETDEV\_VEHICLE\_TYPE\_BUSINESSVEH\_E = 17, /\* MPV \*/

NETDEV\_VEHICLE\_TYPE\_SPORTSCAR\_E = 18, /\* Roadster \*/

NETDEV\_VEHICLE\_TYPE\_MINICAR\_E = 19, /\* Mini car \*/

NETDEV\_VEHICLE\_TYPE\_HATCHBACKCAR\_E = 20, /\* Two-box sedan \*/

NETDEV\_VEHICLE\_TYPE\_THREEBOX\_E = 21, /\* Three-box sedan \*/

NETDEV\_VEHICLE\_TYPE\_LIGHTBUS\_E = 22, /\* Light bus \*/

NETDEV\_VEHICLE\_TYPE\_MEDIUNTRUCK\_E = 23, /\* Medium duty truck \*/

NETDEV\_VEHICLE\_TYPE\_TRAILER\_E = 24, /\* Trailer \*/

NETDEV\_VEHICLE\_TYPE\_TANK\_E = 25, /\* Tanker \*/

NETDEV\_VEHICLE\_TYPE\_WATERCAR\_E = 26, /\* Street sprinkler \*/

NETDEV\_VEHICLE\_TYPE\_OTHER\_E = 998, /\* Other \*/

NETDEV\_VEHICLE\_TYPE\_UNKNOW\_E = 999, /\* Unknown \*/

NETDEV\_VEHICLE\_TYPE\_INDISTINGUISH\_E = 1000, /\* Not distinguish vehicle types \*/

NETDEV\_VEHICLE\_TYPE\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_VEHICLE\_TYPE\_E;

### Enumeration of plate colors

typedef enum tagNETDEVPlateColor

{

NETDEV\_PLATE\_COLOR\_BLACK\_E = 0, /\* Black\*/

NETDEV\_PLATE\_COLOR\_WHITE\_E = 1, /\* White\*/

NETDEV\_PLATE\_COLOR\_GRAY\_E = 2, /\* Grey\*/

NETDEV\_PLATE\_COLOR\_RED\_E = 3, /\* Red\*/

NETDEV\_PLATE\_COLOR\_BLUE\_E = 4, /\* Blue\*/

NETDEV\_PLATE\_COLOR\_YELLOW\_E = 5, /\* Yellow\*/

NETDEV\_PLATE\_COLOR\_ORANGE\_E = 6, /\* Orange\*/

NETDEV\_PLATE\_COLOR\_BROWN\_E = 7, /\* Brown\*/

NETDEV\_PLATE\_COLOR\_GREEN\_E = 8, /\* Green\*/

NETDEV\_PLATE\_COLOR\_PURPLE\_E = 9, /\* Purple\*/

NETDEV\_PLATE\_COLOR\_CYAN\_E = 10, /\* Cyan\*/

NETDEV\_PLATE\_COLOR\_PINK\_E = 11, /\* Pink\*/

NETDEV\_PLATE\_COLOR\_TRANSPARENT\_E = 12, /\* Transparent\*/

NETDEV\_PLATE\_COLOR\_SILVERYWHITE\_E = 13, /\* Silver white\*/

NETDEV\_PLATE\_COLOR\_DARK\_E = 14, /\* Dark\*/

NETDEV\_PLATE\_COLOR\_LIGHT\_E = 15, /\* Light\*/

NETDEV\_PLATE\_COLOR\_COLOURLESS = 16, /\* Colorless\*/

NETDEV\_PLATE\_COLOR\_YELLOWGREEN = 17, /\* Yellow and green \*/

NETDEV\_PLATE\_COLOR\_GRADUALGREEN = 18, /\* Gradual green \*/

NETDEV\_PLATE\_COLOR\_OTHER\_E = 99, /\* Other\*/

NETDEV\_PLATE\_COLOR\_UNKNOW\_E = 100, /\* Unknown\*/

NETDEV\_PLATE\_COLOR\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_PLATE\_COLOR\_E;

### Enumeration of plate types

typedef enum tagNETDEVPlateType

{

NETDEV\_PLATE\_TYPE\_BIG\_CAR\_E = 0, /\* Large vehicle plate \*/

NETDEV\_PLATE\_TYPE\_MINI\_CAR\_E = 1, /\* Small vehicle plate \*/

NETDEV\_PLATE\_TYPE\_EMBASSY\_CAR\_E = 2, /\* Embassy vehicle plate \*/

NETDEV\_PLATE\_TYPE\_CONSULATE\_CAR\_E = 3, /\* Consulate vehicle plate \*/

NETDEV\_PLATE\_TYPE\_OVERSEAS\_CAR\_E = 4, /\* Overseas vehicle plate \*/

NETDEV\_PLATE\_TYPE\_FOREIGN\_CAR\_E = 5, /\* Foreign vehicle plate \*/

NETDEV\_PLATE\_TYPE\_COMMON\_MOTORBIKE\_E = 6, /\* Common motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_HANDINESS\_MOTORBIKE\_E = 7, /\* Light motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_EMBASSY\_MOTORBIKE\_E = 8, /\* Embassy motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_CONSULATE\_MOTORBIKE\_E = 9, /\* Consulate motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_OVERSEAS\_MOTORBIKE\_E = 10, /\* Overseas motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_FOREIGN\_MOTORBIKE\_E = 11, /\* Foreign motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_LOW\_SPEED\_CAR\_E = 12, /\* Low speed vehicle plate \*/

NETDEV\_PLATE\_TYPE\_TRACTOR\_E = 13, /\* Tractor plate \*/

NETDEV\_PLATE\_TYPE\_TRAILER\_E = 14, /\* Trailer plate \*/

NETDEV\_PLATE\_TYPE\_COACH\_CAR\_E = 15, /\* Coach vehicle plate \*/

NETDEV\_PLATE\_TYPE\_COACH\_MOTORBIKE\_E = 16, /\* Coach motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_TEMPORARY\_ENTRY\_CAR\_E = 17, /\* Temporary entry vehicle plate \*/

NETDEV\_PLATE\_TYPE\_TEMPORARY\_ENTRY\_MOTORBIKE\_E = 18, /\* Temporary entry motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_TEMPORARY\_DRIVING\_E = 19, /\* Temporary plate \*/

NETDEV\_PLATE\_TYPE\_POLICE\_CAR\_E = 20, /\* Police vehicle plate \*/

NETDEV\_PLATE\_TYPE\_POLICE\_MOTORBIKE\_E = 21, /\* Police motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_AGRICULTURAL\_E = 22, /\* Agricultural vehicle plate \*/

NETDEV\_PLATE\_TYPE\_HONGKONG\_ENTRY\_EXIT\_E = 23, /\* Border crossing vehicle plate (Hong Kong)\*/

NETDEV\_PLATE\_TYPE\_MACAO\_ENTRY\_EXIT\_E = 24, /\* Border crossing vehicle plate (Macau) \*/

NETDEV\_PLATE\_TYPE\_ARMED\_POLICE\_E = 25, /\* Armed police plate \*/

NETDEV\_PLATE\_TYPE\_ARMY\_E = 26, /\* Military plate \*/

NETDEV\_PLATE\_TYPE\_OTHER\_E = 99, /\* Other \*/

NETDEV\_PLATE\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_PLATE\_TYPE\_E;

### Enumeration of notification types

typedef enum tagNETDEVNotificationType

{

NETDEV\_NOTIFICATION\_TYPE\_REALTIME = 0, /\* Real-time notification \*/

NETDEV\_NOTIFICATION\_TYPE\_HISTORY = 1, /\* Historical notification \*/

NETDEV\_NOTIFICATION\_TYPE\_EARLYWARN = 2, /\* Warning \*/

NETDEV\_NOTIFICATION\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_NOTIFICATION\_TYPE\_E;

### Enumeration of smart alarm types

typedef enum tagNETDEVSmartAlarmType

{

NETDEV\_SMART\_ALARM\_TYPE\_FACE\_SNAP = 0, /\* Face recognition snapshot \*/

NETDEV\_SMART\_ALARM\_TYPE\_VEHICLE\_SNAP = 1, /\* Plate recognition snapshot \*/

NETDEV\_SMART\_ALARM\_TYPE\_VIDEO\_STRUCT\_SNAP = 3, /\* Structured video snapshot \*/

NETDEV\_SMART\_ALARM\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_SMART\_ALARM\_TYPE\_E;

### Enumeration of LAPI subscription types

typedef enum tagNETDEVLAPISubType

{

NETDEV\_LAPI\_SUB\_TYPE\_PARKRECOGNIZE = 2048, /\* Parking lot recognition snapshot \*/

NETDEV\_LAPI\_SUB\_TYPE\_INVALID = 0xffffffff /\* Invalid value \*/

}NETDEV\_LAPI\_SUB\_TYPE\_E;

### Enumeration of lane types

typedef enum tagNETDEVVehLaneType

{

NETDEV\_VEH\_LANE\_TYPE\_ENTRANCE = 0, /\* Entrance \*/

NETDEV\_VEH\_LANE\_TYPE\_EXIT\_NOCHARGE = 1, /\* Exit (do not charge) \*/

NETDEV\_VEH\_LANE\_TYPE\_EXITE\_CHARGE = 2, /\* Exit (charge) \*/

NETDEV\_VEH\_LANE\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_VEH\_LANE\_TYPE\_E;

### Enumeration of vehicle releasing types

typedef enum tagNETDEVVehReleaseType

{

NETDEV\_VEH\_RELEASE\_TYPE\_AUTO = 0, /\* Release automatically \*/

NETDEV\_VEH\_RELEASE\_TYPE\_MANUAL = 1, /\* Release manually \*/

NETDEV\_VEH\_RELEASE\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_VEH\_RELEASE\_TYPE\_E;

### Enumeration of order payment statuses

typedef enum tagNETDEVOrderStatus

{

NETDEV\_ORDER\_STATUS\_FINISH = 0, /\* Unpaid \*/

NETDEV\_ORDER\_STATUS\_UNFINISH = 1, /\* Paid \*/

NETDEV\_ORDER\_STATUS\_PREPAYMENT = 2, /\* Paid in advance \*/

NETDEV\_ORDER\_STATUS\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_ORDER\_STATUS\_E;

### Enumeration of payment types

typedef enum tagNETDEVPayType

{

NETDEV\_PAY\_TYPE\_CASH = 0, /\* Cash \*/

NETDEV\_PAY\_TYPE\_ALIPAY = 1, /\* AliPay \*/

NETDEV\_PAY\_TYPE\_WECHAT = 2, /\* WeChat \*/

NETDEV\_PAY\_TYPE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_PAY\_TYPE\_E;

### Enumeration of person types

typedef enum tagNETDEVACSPersonType

{

NETDEV\_ACS\_PERSON\_TYPE\_STAFF = 0, /\* Staff \*/

NETDEV\_ACS\_PERSON\_TYPE\_VISITOR = 1, /\* Visitor \*/

NETDEV\_ACS\_PERSON\_TYPE\_STRANGER = 2, /\* Stranger \*/

NETDEV\_ACS\_PERSON\_TYPE\_BLACKLIST = 3, /\* Blocklisted people \*/

NETDEV\_ACS\_PERSON\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_ACS\_PERSON\_TYPE\_E;

### Enumeration of visitor statuses

typedef enum tagNETDEVACSVisitStaus

{

NETDEV\_ACS\_VISIT\_STATUS\_SCHEDULE = 0, /\* Scheduled \*/

NETDEV\_ACS\_VISIT\_STATUS\_VISITING = 1, /\* Visiting \*/

NETDEV\_ACS\_VISIT\_STATUS\_LEAVE = 2, /\* Leave \*/

NETDEV\_ACS\_VISIT\_STATUS\_SCHEDULE\_CANCEL = 3, /\* Cancelled \*/

NETDEV\_ACS\_VISIT\_STATUS\_TIMEOUT = 4, /\* Timeout \*/

NETDEV\_ACS\_VISIT\_STATUS\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_ACS\_VISIT\_STATUS\_E;

### Enumeration of collection sources

typedef enum tagNETDEVCapSrc

{

NETDEV\_CAP\_SRC\_FACE = 1, /\* Face info collected by face recognition terminal \*/

NETDEV\_CAP\_SRC\_ENTRANCE\_GUARDCARD = 2, /\* Access control card info collected by card reader \*/

NETDEV\_CAP\_SRC\_ID = 3, /\* ID card info collected by card reader \*/

NETDEV\_CAP\_SRC\_GATE = 4, /\* Gate info collected by gate \*/

NETDEV\_CAP\_SRC\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_CAP\_SRC\_E;

### Enumeration of match statuses

typedef enum tagNETDEVMatchStatus

{

NETDEV\_MATCH\_STATUS\_SUCCESS = 1, /\* Authentication succeeded \*/

NETDEV\_MATCH\_STATUS\_FAIL = 2, /\* Authentication failed (comparison failure) \*/

NETDEV\_MATCH\_STATUS\_NO\_MONITOR\_TIME = 3, /\* Authentication failed (comparison succeeded, not within monitoring time) \*/

NETDEV\_MATCH\_STATUS\_ATTR\_ABNORMAL = 10, /\* Authentication failed (comparison succeeded, abnormal attribute) \*/

NETDEV\_MATCH\_STATUS\_BASE\_MAP\_COLLECT\_SUCC = 41, /\* Background image collected successfully \*/

NETDEV\_MATCH\_STATUS\_BASE\_MAP\_COLLECT\_FAIL = 42, /\* Background image collection failed \*/ NETDEV\_MATCH\_STATUS\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_MATCH\_STATUS\_E;

### Enumeration of image formats

typedef enum tagNETDEVImageFormat

{

NETDEV\_IMAGE\_FORMAT\_JPG = 0, /\* JPG \*/

NETDEV\_IMAGE\_FORMAT\_BMP = 1, /\* BMP \*/

NETDEV\_IMAGE\_FORMAT\_PNG = 2, /\* PNG \*/

NETDEV\_IMAGE\_FORMAT\_GIF = 3, /\* GIF \*/

NETDEV\_IMAGE\_FORMAT\_TIFF = 4, /\* TIFF \*/

NETDEV\_IMAGE\_FORMAT\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_IMAGE\_FORMAT\_E;

### Plate colors

typedef enum tagNETDEVPlateColor

{

NETDEV\_PLATE\_COLOR\_BLACK\_E = 0, /\* Black \*/

NETDEV\_PLATE\_COLOR\_WHITE\_E = 1, /\* White \*/

NETDEV\_PLATE\_COLOR\_GRAY\_E = 2, /\* Grey \*/

NETDEV\_PLATE\_COLOR\_RED\_E = 3, /\* Red \*/

NETDEV\_PLATE\_COLOR\_BLUE\_E = 4, /\* Blue \*/

NETDEV\_PLATE\_COLOR\_YELLOW\_E = 5, /\* Yellow \*/

NETDEV\_PLATE\_COLOR\_ORANGE\_E = 6, /\* Orange \*/

NETDEV\_PLATE\_COLOR\_BROWN\_E = 7, /\* Brown \*/

NETDEV\_PLATE\_COLOR\_GREEN\_E = 8, /\* Green \*/

NETDEV\_PLATE\_COLOR\_PURPLE\_E = 9, /\* Purple \*/

NETDEV\_PLATE\_COLOR\_CYAN\_E = 10, /\* Cyan \*/

NETDEV\_PLATE\_COLOR\_PINK\_E = 11, /\* Pink \*/

NETDEV\_PLATE\_COLOR\_TRANSPARENT\_E = 12, /\* Transparent \*/

NETDEV\_PLATE\_COLOR\_SILVERYWHITE\_E = 13, /\* Silver white \*/

NETDEV\_PLATE\_COLOR\_DARK\_E = 14, /\* Dark \*/

NETDEV\_PLATE\_COLOR\_LIGHT\_E = 15, /\* Light \*/

NETDEV\_PLATE\_COLOR\_COLOURLESS = 16, /\* Colorless \*/

NETDEV\_PLATE\_COLOR\_YELLOWGREEN = 17, /\* Yellow and green \*/

NETDEV\_PLATE\_COLOR\_GRADUALGREEN = 18, /\* Gradient green \*/

NETDEV\_PLATE\_COLOR\_OTHER\_E = 99, /\* Other \*/

NETDEV\_PLATE\_COLOR\_UNKNOW\_E = 100, /\* Unknown \*/

NETDEV\_PLATE\_COLOR\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_PLATE\_COLOR\_E;

### Plate types

typedef enum tagNETDEVPlateType

{

NETDEV\_PLATE\_TYPE\_BIG\_CAR\_E = 0, /\* Large vehicle plate \*/

NETDEV\_PLATE\_TYPE\_MINI\_CAR\_E = 1, /\* Small vehicle plate \*/

NETDEV\_PLATE\_TYPE\_EMBASSY\_CAR\_E = 2, /\* Embassy vehicle plate \*/

NETDEV\_PLATE\_TYPE\_CONSULATE\_CAR\_E = 3, /\* Consulate vehicle plate \*/

NETDEV\_PLATE\_TYPE\_OVERSEAS\_CAR\_E = 4, /\* Overseas vehicle plate \*/

NETDEV\_PLATE\_TYPE\_FOREIGN\_CAR\_E = 5, /\* Foreign vehicle plate \*/

NETDEV\_PLATE\_TYPE\_COMMON\_MOTORBIKE\_E = 6, /\* Common motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_HANDINESS\_MOTORBIKE\_E = 7, /\* Light motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_EMBASSY\_MOTORBIKE\_E = 8, /\* Embassy motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_CONSULATE\_MOTORBIKE\_E = 9, /\* Consulte motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_OVERSEAS\_MOTORBIKE\_E = 10, /\* Overseas motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_FOREIGN\_MOTORBIKE\_E = 11, /\* Foreign motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_LOW\_SPEED\_CAR\_E = 12, /\* Low speed vehicle plate \*/

NETDEV\_PLATE\_TYPE\_TRACTOR\_E = 13, /\* Tractor plate \*/

NETDEV\_PLATE\_TYPE\_TRAILER\_E = 14, /\* Trailer plate \*/

NETDEV\_PLATE\_TYPE\_COACH\_CAR\_E = 15, /\* Coach vehicle plate \*/

NETDEV\_PLATE\_TYPE\_COACH\_MOTORBIKE\_E = 16, /\* Coach motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_TEMPORARY\_ENTRY\_CAR\_E = 17, /\* Temporary entry vehicle plate \*/

NETDEV\_PLATE\_TYPE\_TEMPORARY\_ENTRY\_MOTORBIKE\_E = 18, /\* Temporary entry motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_TEMPORARY\_DRIVING\_E = 19, /\* Temporary plate \*/

NETDEV\_PLATE\_TYPE\_POLICE\_CAR\_E = 20, /\* Police vehicle plate \*/

NETDEV\_PLATE\_TYPE\_POLICE\_MOTORBIKE\_E = 21, /\* Police motorcycle plate \*/

NETDEV\_PLATE\_TYPE\_AGRICULTURAL\_E = 22, /\* Agricultural vehicle plate \*/

NETDEV\_PLATE\_TYPE\_HONGKONG\_ENTRY\_EXIT\_E = 23, /\* Border crossing vehicle plate (Hong Kong) \*/

NETDEV\_PLATE\_TYPE\_MACAO\_ENTRY\_EXIT\_E = 24, /\* Border crossing vehicle plate (Macau) \*/

NETDEV\_PLATE\_TYPE\_ARMED\_POLICE\_E = 25, /\* Armed police plate \*/

NETDEV\_PLATE\_TYPE\_ARMY\_E = 26, /\* Military plate \*/

NETDEV\_PLATE\_TYPE\_OTHER\_E = 99, /\* Other \*/

NETDEV\_PLATE\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_PLATE\_TYPE\_E;

### Vehicle types

typedef enum tagNETDEVVehicleType

{

NETDEV\_VEHICLE\_TYPE\_TRICYCLE\_E = 0, /\* 3-wheel vehicle \*/

NETDEV\_VEHICLE\_TYPE\_MOTOR\_BUS\_E = 1, /\* Bus \*/

NETDEV\_VEHICLE\_TYPE\_MIDDLE\_E = 2, /\* Mid-size vehicle \*/

NETDEV\_VEHICLE\_TYPE\_SMALL\_E = 3, /\* Small vehicle \*/

NETDEV\_VEHICLE\_TYPE\_BIG\_E = 4, /\* Large vehicle \*/

NETDEV\_VEHICLE\_TYPE\_TWOWHEELVEH = 5, /\* 2-wheel vehicle \*/

NETDEV\_VEHICLE\_TYPE\_MOTORCYCLE\_E = 6, /\* Motorcycle \*/

NETDEV\_VEHICLE\_TYPE\_TRACTOR\_E = 7, /\* Tractor \*/

NETDEV\_VEHICLE\_TYPE\_AGRICULTURAL\_E = 8, /\* Farm truck \*/

NETDEV\_VEHICLE\_TYPE\_SEADAN = 9, /\* Sedan \*/

NETDEV\_VEHICLE\_TYPE\_SUV\_E = 10, /\* SUV \*/

NETDEV\_VEHICLE\_TYPE\_VAN\_E = 11, /\* Minibus \*/

NETDEV\_VEHICLE\_TYPE\_SMALLTRUCK\_E = 12, /\* Van \*/

NETDEV\_VEHICLE\_TYPE\_MEDIUMCAR\_E = 13, /\* Mid-sized bus/medium bus \*/

NETDEV\_VEHICLE\_TYPE\_LARGEBUS\_E = 14, /\* Large bus \*/

NETDEV\_VEHICLE\_TYPE\_LARGETRUCK\_E = 15, /\* Large truck \*/

NETDEV\_VEHICLE\_TYPE\_PICKUPTRUCK\_E = 16, /\* Pickup \*/

NETDEV\_VEHICLE\_TYPE\_BUSINESSVEH\_E = 17, /\* MPV \*/

NETDEV\_VEHICLE\_TYPE\_SPORTSCAR\_E = 18, /\* Roadster \*/

NETDEV\_VEHICLE\_TYPE\_MINICAR\_E = 19, /\* Mini car \*/

NETDEV\_VEHICLE\_TYPE\_HATCHBACKCAR\_E = 20, /\* Two-box sedan \*/

NETDEV\_VEHICLE\_TYPE\_THREEBOX\_E = 21, /\* Three-box sedan \*/

NETDEV\_VEHICLE\_TYPE\_LIGHTBUS\_E = 22, /\* Light bus \*/

NETDEV\_VEHICLE\_TYPE\_MEDIUNTRUCK\_E = 23, /\* Medium duty truck \*/

NETDEV\_VEHICLE\_TYPE\_TRAILER\_E = 24, /\* Trailer \*/

NETDEV\_VEHICLE\_TYPE\_TANK\_E = 25, /\* Tanker \*/

NETDEV\_VEHICLE\_TYPE\_WATERCAR\_E = 26, /\* Street sprinkler \*/

NETDEV\_VEHICLE\_TYPE\_OTHER\_E = 998, /\* Other \*/

NETDEV\_VEHICLE\_TYPE\_UNKNOW\_E = 999, /\* Unknown \*/

NETDEV\_VEHICLE\_TYPE\_INDISTINGUISH\_E = 1000, /\* Not distinguish vehicle types \*/

NETDEV\_VEHICLE\_TYPE\_INVALID = 0xFFFF /\* Invalid value Invalid value \*/

}NETDEV\_VEHICLE\_TYPE\_E;

### Enumeration of speed types of non-motor vehicles in structured scenes

typedef enum tagNETDEVSpeedType

{

NETDEV\_SPEED\_TYPE\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_SPEED\_TYPE\_STATIC = 1, /\* Static \*/

NETDEV\_SPEED\_TYPE\_SLOW = 2, /\* Slow \*/

NETDEV\_SPEED\_TYPE\_MEDIUM = 3, /\* Medium \*/

NETDEV\_SPEED\_TYPE\_FAST = 4, /\* Fast \*/

NETDEV\_SPEED\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_SPEED\_TYPE\_E;

### Enumeration of movement directions (relative to image) of non-motor vehicles in structured scenes

typedef enum tagNETDEVImageDirection

{

NETDEV\_IMAGE\_DIRECTION\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_IMAGE\_DIRECTION\_STATIC = 1, /\* Static \*/

NETDEV\_IMAGE\_DIRECTION\_UP = 2, /\* Up \*/

NETDEV\_IMAGE\_DIRECTION\_DOWN = 3, /\* Down \*/

NETDEV\_IMAGE\_DIRECTION\_LEFT = 4, /\* Left \*/

NETDEV\_IMAGE\_DIRECTION\_RIGHT = 5, /\* Right \*/

NETDEV\_IMAGE\_DIRECTION\_LEFTUP = 6, /\* Up left \*/

NETDEV\_IMAGE\_DIRECTION\_LEFTDOWN = 7, /\* Down left \*/

NETDEV\_IMAGE\_DIRECTION\_RIGHTUP = 8, /\* Up right \*/

NETDEV\_IMAGE\_DIRECTION\_RIGHTDOWN = 9, /\* Down right \*/

NETDEV\_IMAGE\_DIRECTION\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_IMAGE\_DIRECTION\_E;

### Enumeration of channel permissions

typedef enum tagNETDEVChnPermission

{

NETDEV\_CHN\_PERMISSION\_LIVE = BIT0, /\* Live view and two-way audio \*/

NETDEV\_CHN\_PERMISSION\_PTZ = BIT1, /\* PTZ control \*/

NETDEV\_CHN\_PERMISSION\_PLAYBACK = BIT2, /\* Playback \*/

NETDEV\_CHN\_PERMISSION\_MANU\_RECORD = BIT3, /\* Manual recording \*/

NETDEV\_CHN\_PERMISSION\_LOCAL\_BACK = BIT4, /\* Local backup \*/

NETDEV\_CHN\_PERMISSION\_INVALID = 0XFFFFFFFF /\* Invalid value \*/

}NETDEV\_CHN\_PERMISSION\_TYPE\_E;

### Enumeration of response statuses

typedef enum tagNETDEVOrgResponseStatus

{

NETDEV\_ORG\_RESPONSE\_SUCCESS = 0, /\* Success \*/

NETDEV\_ORG\_RESPONSE\_FAIL = 1 /\* Fail \*/

}NETDEV\_ORG\_RESPONSE\_STAUTE\_E;

### Enumeration of NAT types

typedef enum tagNETDEVNATType

{

NETDEV\_STUN\_TYPE\_UNKNOWN = 0, /\* Unknown \*/

NETDEV\_STUN\_TYPE\_FAILURE = 1, /\* Failure \*/

NETDEV\_STUN\_TYPE\_OPEN = 2 /\* No NAT; public IP, no firewall \*/

NETDEV\_STUN\_TYPE\_BLOCKED = 3, /\* UDP blocked \*/

NETDEV\_STUN\_TYPE\_CONE\_NAT = 4, /\* Full cone NAT \*/

NETDEV\_STUN\_TYPE\_RESTRICTED\_NAT = 5, /\* Restricted cone NAT \*/

NETDEV\_STUN\_TYPE\_PORT\_RESTRICTED\_NAT = 6, /\* Port restricted NAT \*/

NETDEV\_STUN\_TYPE\_SYM\_NAT = 7, /\* Symmetric NAT \*/

NETDEV\_STUN\_TYPE\_SYM\_FIREWALL = 8 /\* No NAT; public IP, UDP symmetric firewall \*/

}NETDEV\_NAT\_TYPE\_E;

### Enumeration of protocol types

typedef enum tagNETDEVProtocolType

{

NETDEV\_PROTOCOL\_TYPE\_HTTP = 0,

NETDEV\_PROTOCOL\_TYPE\_HTTPS = 1,

NETDEV\_PROTOCOL\_TYPE\_RTSP = 2

}NETDEV\_PROTOCOL\_TYPE\_E;

### Enumeration of organization search modes

typedef enum tagNETDEVOrgFindMode

{

NETDEV\_ORG\_FIND\_MODE\_ID = 0, /\* Organization ID \*/

NETDEV\_ORG\_FIND\_MODE\_TYPE = 1, /\* Organization type \*/

NETDEV\_ORG\_FIND\_MODE\_INVALID = 0XFF /\* Invalid value \*/

}NETDEV\_ORG\_FIND\_MODE\_E;

### Enumeration of basic permissions

typedef enum tagNETDEVUserBasePermission

{

NETDEV\_USER\_BASE\_PER\_CONFIG = BIT0, /\* Configure \*/

NETDEV\_USER\_BASE\_PER\_UPGRADE = BIT1, /\* Upgrade \*/

NETDEV\_USER\_BASE\_PER\_LOG = BIT2, /\* View and export log \*/

NETDEV\_USER\_BASE\_PER\_REBOOT = BIT3 /\* Restart \*/

}NETDEV\_USER\_BASE\_PERMISSION\_E;

### HDD modes

typedef enum tagNETDEVDiskModeType

{

NETDEV\_DISK\_MODE\_TYPE\_OLD = 0, /\* Old storage format disk \*/

NETDEV\_DISK\_MODE\_TYPE\_NEW = 1, /\* New storage format disk \*/

NETDEV\_DISK\_MODE\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_DISK\_MODE\_TYPE\_E;

### Capability commands

typedef enum tagNETDEVCapabilityCommond

{

NETDEV\_CAP\_VIDEO\_ENCODE = 1, /\* Video encoding capability. See

NETDEV\_VIDEO\_STREAM\_CAP\_S \*/

NETDEV\_CAP\_OSD = 2, /\* OSD parameter capability. See # NETDEV\_OSD\_CAP\_S \*/

NETDEV\_CAP\_SMART = 3, /\* VCA capability. See # NETDEV\_SMART\_CAP\_S \*/

NETDEV\_CAP\_VIDEO\_ENCODE\_EX = 4, /\* Video encoding capability (expansion). See #

NETDEV\_VIDEO\_STREAM\_CAP\_EX\_S \*/

NETDEV\_CAP\_IMAGE = 5, /\* Image parameter capability. See #NETDEV\_IMAGE\_CAP\_S \*/

NETDEV\_CAP\_AUDIO = 6, /\* Audio capability. See NETDEV\_AUDIO\_CAP\_S \*/

NETDEV\_CAP\_VIDEO\_SNAPSHOT = 7, /\* Video input channel’s snapshot capability. See NETDEV\_VIDEO\_SNAP\_CAP\_S. Currently supported by IPC only. \*/

NETDEV\_CAP\_FACE\_RECOGNAZE = 10, /\* Face recognition capability. Currently supported by NVR only. See NETDEV\_DEV\_FACE\_CAP\_INFO\_S \*/

NETDEV\_CAP\_DEV\_NIC = 11, /\* Device’s NIC capability. See

NETDEV\_NIC\_CAP\_INFO\_S \*/

NETDEV\_CAP\_DEV\_EXCEPTION\_ALARM = 12, /\* Device’s exception alarm capability. See

NETDEV\_EXCP\_ALARM\_CAP\_INFO\_S \*/

NETDEV\_CAP\_CHANNELS\_ALARM = 13, /\* Channel's alarm capability. See

NETDEV\_CHN\_ALARM\_CAP\_INFO\_S (For single-channel IPC, it corresponds to the passed-in channel ID; for multi-channel IPC, it corresponds to the passed-in channel ID + actual channel ID of IPC; for an NVR channel, it corresponds to the actual passed-in channel ID) \*/

NETDEV\_CAP\_SYS = 14, /\* System capability. See

NETDEV\_SYS\_CAPABILITY\_S \*/

NETDEV\_CAP\_INVALID = 0Xff

}NETDEV\_CAPABILITY\_COMMOND\_E;

### OSD date format capability

typedef enum tagNETDEVDateOSDFormateCap

{

NETDEV\_OSD\_DATE\_FORMAT\_CAP\_MD\_YYYY = 0, /\* M/d/yyyy \*/

NETDEV\_OSD\_DATE\_FORMAT\_CAP\_MMDD\_YYYY = 1, /\* MM/dd/yyyy \*/

NETDEV\_OSD\_DATE\_FORMAT\_CAP\_DDMM\_YYYY = 2, /\* dd/MM/yyyy \*/

NETDEV\_OSD\_DATE\_FORMAT\_CAP\_YYYY\_MMDD = 3, /\* yyyy/MM/dd \*/

NETDEV\_OSD\_DATE\_FORMAT\_CAP\_YYYYMMDDB = 4, /\* yyyy-MM-dd \*/

NETDEV\_OSD\_DATE\_FORMAT\_CAP\_XX\_MMDD\_YYYY = 5, /\* dddd, MMMM dd, yyyy \*/

NETDEV\_OSD\_DATE\_FORMAT\_CAP\_MMMMDD\_YYYY = 6, /\* MMMM dd, yyyy \*/

NETDEV\_OSD\_DATE\_FORMAT\_CAP\_DDMMMM\_YYYY = 7 /\* dd MMMM, yyyy \*/

}NETDEV\_OSD\_DATE\_FORMAT\_CAP\_E;

### Enumeration of time sync modes

typedef enum tagNETDEVTimeSynmode

{

NETDEV\_TIME\_SYNCMODE\_ACCEPT\_ALL\_SERVER = 0, /\* Sync with all servers \*/

NETDEV\_TIME\_SYNCMODE\_SYNC\_COMPUTER\_SYSTEM = 1, /\* Sync with PC’s system time \*/

NETDEV\_TIME\_SYNCMODE\_SYNC\_PHOTO\_SERVER = 2, /\* Sync with photo server \*/

NETDEV\_TIME\_SYNCMODE\_SYNC\_NTP\_SERVER = 3, /\* Sync with NTP server \*/

NETDEV\_TIME\_SYNCMODE\_SYNC\_MANAGE\_SERVER = 4, /\* Sync with management server (non-Onvif) \*/

NETDEV\_TIME\_SYNCMODE\_SYNC\_MANAGE\_SERVER\_ONVIF= 5,/\* Sync with management server (Onvif) \*/

NETDEV\_TIME\_SYNCMODE\_SYNC\_CLOUD\_SERVER = 6, /\* Sync with cloud server \*/

NETDEV\_TIME\_SYNCMODE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_TIME\_SYNCMODE\_E;

### Stream data callback function

typedef enum tagNETDEStreamDataCBType

{

NETDEV\_STREAM\_CB\_TYPE\_PARSE = 1, /\* Callback of assembled audio/video data \*/

NETDEV\_STREAM\_CB\_TYPE\_DECODE = 2, /\* Callback of decoded audio/video data \*/

NETDEV\_STREAM\_CB\_TYPE\_DECODE\_V30 = 3, /\* Callback of decoded audio/video data. GPU decoding is supported \*/

NETDEV\_STREAM\_CB\_TYPE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_STREAM\_DATA\_CB\_TYPE\_E;

### People counting types

typedef enum tagNETDEVStatisticType

{

NETDEV\_STATISTIC\_TYPE\_ALL = 0, /\* Count the total number of people

NETDEV\_STATISTIC\_TYPE\_ENTER = 1, /\* Count people who have entered \*/

NETDEV\_STATISTIC\_TYPE\_LEAVE = 2, /\* Count people who have left \*/

NETDEV\_STATISTIC\_TYPE\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_STATISTIC\_TYPE\_E;

### Enumeration of listening types

typedef enum tagNETDEVListenType

{

NETDEV\_FACE\_SNAPSHOT\_TYPE = 1, /\* Face snapshot \*/

NETDEV\_DISCOVERY\_TYPE = 2, /\* Device discovery \*/

NETDEV\_LISTEN\_TYPE\_INVALID = 0xffff

}NETDEV\_LISTEN\_TYPE\_E;

### Enumeration of supported alarm types

typedef enum tagNETDEVAlarmTypeV30

{

NETDEV\_ALARM\_RYPE\_DEV\_STATUS = BIT0, /\* Device status alarm \*/

NETDEV\_ALARM\_RYPE\_COMM\_ALARM = BIT1, /\* Surveillance alarm \*/

NETDEV\_ALARM\_RYPE\_INTEL\_ALARM = BIT2, /\* Ordinary smart alarm \*/

NETDEV\_ALARM\_RYPE\_SMART\_ALARM = BIT3, /\* Deep learning smart alarm \*/

NETDEV\_ALARM\_RYPE\_FACE\_RECOGNITION = BIT4, /\* Face recognition NETDEV\_SetPersonAlarmCallBack \*/

NETDEV\_ALARM\_RYPE\_STRUCTURED\_DATA = BIT5, /\* Structured data NETDEV\_SetStructAlarmCallBack \*/

NETDEV\_ALARM\_RYPE\_VEHICLE\_RECOGNITION = BIT6, /\* Plate recognition NETDEV\_SetVehicleAlarmCallBack \*/

NETDEV\_ALARM\_RYPE\_TRAFFIC\_DATA = BIT7, /\* Traffic data (currently unavailable) \*/

NETDEV\_ALARM\_RYPE\_HYPERSENSITIVE\_DATA = BIT8, /\* Currently not available \*/

NETDEV\_ALARM\_RYPE\_RESOURCE\_CHANGE = BIT9, /\* Resource change NETDEV\_SetResChangeEventCallBack \*/

NETDEV\_ALARM\_RYPE\_PERSON\_VERIFICATION = BIT10, /\* Face verification NETDEV\_SetAlarmFGCallBack \*/

NETDEV\_ALARM\_RYPE\_PARKING\_IDENTIFICATION = BIT11, /\* Parking lot snapshot NETDEV\_SetParkEventCallBack \*/

NETDEV\_ALARM\_RYPE\_FIREALARM\_DATA = BIT12, /\* Fire detection alarm \*/

NETDEV\_ALARM\_RYPE\_ALARM\_PICTURE\_DATA = BIT13, /\* Alarm image data NETDEV\_SetPicAlarmCallBack \*/

NETDEV\_ALARM\_RYPE\_PEOPLE\_COUNT = BIT14, /\* People counting NETDEV\_SetPeopleCountAlarmCallBack \*/

NETDEV\_ALARM\_RYPE\_HEATMAP\_DATA = BIT16, /\* Heatmap data (currently unavailable) \*/

NETDEV\_ALARM\_RYPE\_PLAYBOX\_STATUS = BIT17, /\* Player status (currently unavailable) \*/

NETDEV\_ALARM\_RYPE\_PLAYBOX\_MANAGEMENT = BIT18, /\* Player manager online status (currently unavailable).\*/

NETDEV\_ALARM\_RYPE\_INVALID = 0xFF

}NETDEV\_ALARM\_TYPE\_V30\_E;

### Enumeration of plate colors

typedef enum tagNETDEVTMSCarPlateColor

{

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_WHITE = 0, /\* White \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_YELLOW = 1, /\* Yellow \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_BLUE = 2, /\* Blue \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_BLACK = 3, /\* Black \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_OTHER = 4, /\* Other colors \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_GREEN = 5, /\* Green, farm vehicles \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_RED = 6, /\* Red \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_YELLOW\_AND\_GREEN = 7, /\* Yellow & green \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_GRADIENT\_GREEN = 8， /\* Gradient green \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_TMS\_CAR\_PLATE\_COLOR\_E;

### Enumeration of recording search types

typedef enum tagNETDEVRecordSearchType

{

NETDEV\_RECORD\_SEARCH\_TYPE\_ALL = 0x00000000, /\* Used in searching, all recording types \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_MANUL = 0x00000001, /\* Manual recording \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_EVENT = 0x00000002, /\* Common alarm (not supported by NVR) \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_MOTION = 0x00000004, /\* Motion detection \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_ALARMIN = 0x00000008, /\* Input alarm \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_VIDEO\_LOSS= 0x00000010, /\* Video loss \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_AUDIO\_DETECT= 0x00000020, /\* Audio detection \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_COMMON = 0x00000080, /\* Scheduled recording (normal) \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_FACE\_DETECT = 0x00000100, /\* Face detection \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_LINE\_DETECT = 0x00000200, /\* Cross line detection \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_FIELD\_DETECT = 0x00000400, /\* Intrusion detection \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_FOCUS\_DETECT = 0x00000800, /\* Defocus detection \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_SCENE\_CHANGE= 0x00001000, /\* Scene change \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_SMART\_TRACK = 0x00002000, /\* Auto track \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_URGENT\_BELL = 0x00004000, /\* Emergency bell \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_REMAIN\_ARTICLE = 0x00020000, /\* Object left behind \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_MOVE\_ARTICLE = 0x00040000, /\* Object removed \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_HUMAN\_DETECT = 0x00200000, /\* Human body detection \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_ELEVATOR\_ENTRANCE\_DETECT= 0x20000000, /\* Elevator entrance detection \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_SMART\_RECORD = 0x20263F20, /\* All smart alarms 0010 0000 0010 0110 0011 1111 0010 0000\*/

NETDEV\_RECORD\_SEARCH\_TYPE\_INVALID = 0XFFFFFFFF /\* Invalid value \*/

}NETDEV\_RECORD\_SEARCH\_TYPE\_E;

### Enumeration of skin tones

typedef enum tagNETDEVSkinColorType

{

NETDEV\_SKINCOLOR\_TYPE\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_SKINCOLOR\_TYPE\_WHITE = 2011, /\* White \*/

NETDEV\_SKINCOLOR\_TYPE\_BLACK = 2012, /\* Black \*/

NETDEV\_SKINCOLOR\_TYPE\_YELLOW = 2013, /\* Yellow \*/

NETDEV\_SKINCOLOR\_TYPE\_BROWN = 2014, /\* Brown \*/

NETDEV\_SKINCOLOR\_TYPE\_INVALID = 0xFF /\* Invalid \*/

}NETDEV\_SKINCOLOR\_TYPE\_E;

### Emotions

typedef enum tagNETDEVEmotionFlag

{

NETDEV\_EMOTION\_FLAG\_UNKNOW = 0, /\* Unknow \*/

NETDEV\_EMOTION\_FLAG\_ANGER = 1, /\* Angry \*/

NETDEV\_EMOTION\_FLAG\_CALM = 2, /\* Calm \*/

NETDEV\_EMOTION\_FLAG\_CONFUSED = 3, /\* Confused \*/

NETDEV\_EMOTION\_FLAG\_ABHORRENT = 4, /\* Abhorrent \*/

NETDEV\_EMOTION\_FLAG\_HAPPY = 5, /\* Happy \*/

NETDEV\_EMOTION\_FLAG\_SAD = 6, /\* Sad \*/

NETDEV\_EMOTION\_FLAG\_AFRAID = 7, /\* Afraid \*/

NETDEV\_EMOTION\_FLAG\_AMAZED = 8, /\* Surprised \*/

NETDEV\_EMOTION\_FLAG\_SQUINT = 9, /\* Squint \*/

NETDEV\_EMOTION\_FLAG\_SCREAM = 10, /\* Screaming \*/

NETDEV\_EMOTION\_FLAG\_OTHER = 11, /\* Other \*/

NETDEV\_EMOTION\_FLAG\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_EMOTION\_FLAG\_E;

### Smile flag

typedef enum tagNETDEVSmileFlag

{

NETDEV\_SMILE\_FLAG\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_SMILE\_FLAG\_NO = 1, /\* No smile \*/

NETDEV\_SMILE\_FLAG\_YES = 2, /\* With smile \*/

NETDEV\_SMILE\_FLAG\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_SMILE\_FLAG\_E;

### Beard flag

typedef enum tagNETDEVBeardFlag

{

NETDEV\_BEARD\_FLAG\_UNKNOW = 0, /\* Unknown \*/

NETDEV\_BEARD\_FLAG\_UNEXIST = 1, /\* No beard \*/

NETDEV\_BEARD\_FLAG\_EXIST = 2, /\* With beard \*/

NETDEV\_BEARD\_FLAG\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_BEARD\_FLAG\_E;

### Enumeration of object types

typedef enum tagNETDEVObjectType

{

NETDEV\_OBJECT\_TYPE\_FACE = 1, /\* Face \*/

NETDEV\_OBJECT\_TYPE\_PERSON = 2, /\* Person \*/

NETDEV\_OBJECT\_TYPE\_NON\_MOTOR = 3, /\* Non-motor vehicle \*/

NETDEV\_OBJECT\_TYPE\_MOTOR = 4, /\* Motor vehicle \*/

NETDEV\_OBJECT\_TYPE\_UNKNOWN = 255, /\* Unknown/unidentified \*/

NETDEV\_OBJECT\_TYPE\_INVALID = 0xffff /\* Invalid value \*/

}NETDEV\_OBJECT\_TYPE\_E;

### Enumeration of rule types

typedef enum tagNETDEVRuleType

{

NETDEV\_RULE\_TYPE\_INTRUSION\_DETECTION = 0, /\* Intrusion detection \*/

NETDEV\_RULE\_TYPE\_CROSSLINE\_DETECTION = 1, /\* Cross line detection \*/

NETDEV\_RULE\_TYPE\_LEAVE\_ZONE = 2, /\* Leave area \*/

NETDEV\_RULE\_TYPE\_ENTER\_ZONE = 3, /\* Enter area \*/

NETDEV\_RULE\_TYPE\_ELEVATOR\_ENTRANCE\_DETECTION = 4, /\* Elevator entrance detection \*/

NETDEV\_RULE\_TYPE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_RULE\_TYPE\_E;

### Enumeration of rule trigger types

typedef enum tagNETDEVRuleTriggerType

{

NETDEV\_RULE\_TRIGGER\_TYPE\_NO\_TRIGGER = 0, /\* No trigger \*/

NETDEV\_RULE\_TRIGGER\_TYPE\_DUPLEX = 1, /\* Bi-directional \*/

NETDEV\_RULE\_TRIGGER\_TYPE\_CLOCKWISE = 2, /\* Clockwise \*/

NETDEV\_RULE\_TRIGGER\_TYPE\_ANTIC\_CLOCKWISE = 3, /\* Counterclockwise \*/

NETDEV\_RULE\_TRIGGER\_TYPE\_ACCESS = 4, /\* Enter \*/

NETDEV\_RULE\_TRIGGER\_TYPE\_LEAVE = 5, /\* Leave \*/

NETDEV\_RULE\_TRIGGER\_TYPE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_RULE\_TRIGGER\_TYPE\_E;

### Target record types

typedef enum tagNETDEVObjectRecordType

{

NETDEV\_OBJECT\_RECORD\_TYPE\_CROSSLINE\_DETECTION = 0, /\* Cross line detection \*/

NETDEV\_OBJECT\_RECORD\_TYPE\_INTRUSION\_DETECTION = 1, /\* Intrusion detection \*/

NETDEV\_OBJECT\_RECORD\_TYPE\_ENTER\_ZONE = 2, /\* Enter area \*/

NETDEV\_OBJECT\_RECORD\_TYPE\_LEAVE\_ZONE = 3, /\* Leave area \*/

NETDEV\_OBJECT\_RECORD\_TYPE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_OBJECT\_RECORD\_TYPE\_E;

### People counting alarm types

typedef enum tagNETDEVPeopleCountAlarmType

{

NETDEV\_PEOPLE\_COUNT\_ALARM\_TYPE\_AREA\_RULE = 0, /\* area people counting \*/

NETDEV\_PEOPLE\_COUNT\_ALARM\_TYPE\_LINE\_RULE = 1, /\* Tripwire people counting \*/

NETDEV\_PEOPLE\_COUNT\_ALARM\_TYPE\_CROWD\_DENSITY = 2, /\* Crowd density \*/

NETDEV\_PEOPLE\_COUNT\_ALARM\_TYPE\_INVALID = 0xFFFF /\* Invalid value \*/

}NETDEV\_PEOPLE\_COUNT\_ALARM\_TYPE\_E;

### Crowd density alarm statistics type

typedef enum tagNETDEVCrowdDensityStatisticType

{

NETDEV\_CROWD\_DENSITY\_STATISTIC\_TYPE\_DENSITY = 1, /\* Crowd density statistics \*/

NETDEV\_CROWD\_DENSITY\_STATISTIC\_TYPE\_STRANDED = 2, /\* Number of people present \*/

NETDEV\_CROWD\_DENSITY\_STATISTIC\_TYPE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_CROWD\_DENSITY\_STATISTIC\_TYPE\_E;

### Temperature alarm value types

typedef enum tagNETDEVTemperatureType

{

NETDEV\_TEMPERATURE\_TYPE\_MAX = 1, /\* Highest temperature \*/

NETDEV\_TEMPERATURE\_TYPE\_MIN = 2, /\* Lowest temperature \*/

NETDEV\_TEMPERATURE\_TYPE\_AVERAGE = 3, /\* Average temperature \*/

NETDEV\_TEMPERATURE\_TYPE\_DIFFERENCE = 4, /\* Temperature difference \*/

NETDEV\_TEMPERATURE\_TYPE\_CHANGE\_RATE = 5, /\* Temperature change rate \*/

NETDEV\_TEMPERATURE\_TYPE\_INVALID = 0xff /\* Invalid value \*/

}NETDEV\_TEMPERATURE\_TYPE\_E;

## Error Code List

### Common Error Codes

|  |  |  |
| --- | --- | --- |
| **Code** | **Value** | **Meaning** |
| NETDEV\_E\_FAILED | -1 | Failure |
| NETDEV\_E\_SUCCEED | 0 | Success |
| NETDEV\_E\_SVC\_FAILED | 1 | Server Error Code |
| NETDEV\_E\_NO\_USER | 5 | User does not exist |
| NETDEV\_E\_NO\_RESULT | 11 | No search results |
| NETDEV\_E\_NOENOUGH\_BUF | 12 | Buffer is too small |
| NETDEV\_E\_SDK\_SOCKET\_LSN\_FAIL | 13 | Failed to create socket listener |
| NETDEV\_E\_INIT\_MUTEX\_FAIL | 14 | Failed to initialize lock |
| NETDEV\_E\_INIT\_SEMA\_FAIL | 15 | Failed to initialize semaphore |
| NETDEV\_E\_ALLOC\_RESOURCE\_ERROR | 16 | SDK resource allocation error |
| NETDEV\_E\_HAVEDATA | 17 | Still have data |
| NETDEV\_E\_NEEDMOREDATA | 18 | More data required |
| NETDEV\_E\_TRANSFILE\_FAIL | 19 | File transfer failed |
| NETDEV\_E\_DEVICE\_TYPE\_ERR | 20 | Unsupported device type |
| NETDEV\_E\_NONCE\_TIMEOUT | 21 | Nonce expired |
| NETDEV\_E\_INNER\_ERR | 22 | System internal error |
| NETDEV\_E\_PUBLICKEYFAIL | 23 | Failed to pair public key |
| NETDEV\_E\_BINDNOTIFY\_FAIL | 24 | Failed to bind alarm |
| NETDEV\_E\_SYSCALL\_FALIED | 100 | Failed to call system function |
| NETDEV\_E\_NULL\_POINT | 101 | Null pointer |
| NETDEV\_E\_INVALID\_PARAM | 102 | Invalid parameter |
| NETDEV\_E\_INVALID\_MODULEID | 103 | Invalid module ID |
| NETDEV\_E\_INVALID\_HANDLE | 104 | Invalid handle |
| NETDEV\_E\_NO\_MEMORY | 105 | Failed to allocate memory |
| NETDEV\_E\_FILE\_NO\_EXIST | 106 | File does not exist |
| NETDEV\_E\_NO\_DEV | 107 | Device does not exist |
| NETDEV\_E\_NO\_FIT\_LOG | 108 | No matches found. |
| NETDEV\_E\_BUSY | 109 | Busy status |
| NETDEV\_E\_TIMER\_REG\_FAILED | 110 | Failed to register timer |
| NETDEV\_E\_COMMON\_FAILED | 111 | Common error |
| NETDEV\_E\_CMD\_NOT\_SUPPORT | 112 | Unsupported command |
| NETDEV\_E\_NOT\_SUPPORT | 113 | Device does not support the function |
| NETDEV\_E\_TIMEOUT | 114 | Timeout |
| NETDEV\_E\_MSG\_ERR | 115 | Message does not match |
| NETDEV\_E\_MODULE\_INEXIST | 116 | Module does not exist |
| NETDEV\_E\_SOCKET\_RECV\_ERR | 117 | Failed to receive messages |
| NETDEV\_E\_DECODE\_IE\_FAILED | 118 | Failed to get message IE |
| NETDEV\_E\_ENCODE\_IE\_FAILED | 119 | Failed to add message IE |
| NETDEV\_E\_SDK\_NOINTE\_ERROR | 120 | SDK is not initialized |
| NETDEV\_E\_ALREDY\_INIT\_ERROR | 121 | SDK is already initialized |
| NETDEV\_E\_DEVICE\_FACTURER\_ERR | 122 | Unsupported device vendor |
| NETDEV\_E\_NAME\_EXIST | 123 | The name already exists |
| NETDEV\_E\_GET\_CFG\_FAILED | 124 | Error occurred when getting configuration info |
| NETDEV\_E\_SET\_CFG\_FAILED | 125 | Error occurred while setting configurations |
| NETDEV\_E\_CHANNEL\_OVER\_SPEC | 126 | Channel number exceeds specifications |
| NETDEV\_E\_CALL\_DRV\_COMMON | 127 | Failed to call driver |
| NETDEV\_E\_TOTAL\_QUOTA\_FULL | 128 | Not enough free space can be allocated |
| NETDEV\_E\_CALL\_DB\_COMMON | 129 | Failed to call database |
| NETDEV\_E\_NEED\_MORE\_MEMORY | 130 | Insufficient memory allocated |
| NETDEV\_E\_T2U\_CONNECT\_FAILED | 131 | T2U connection failed |
| NETDEV\_E\_FUNC\_IS\_INITIALIZING | 132 | Function is initializing |
| NETDEV\_E\_CONNECT\_ERROR | 200 | Failed to establish a connection |
| NETDEV\_E\_SEND\_MSG\_ERROR | 201 | Failed to send message |
| NETDEV\_E\_DECODE\_RSP\_ERROR | 202 | Failed to parse response message |
| NETDEV\_E\_NONSUPPORT | 203 | Unsupported function |
| NETDEV\_E\_JSON\_ERROR | 204 | Common Json error |
| NETDEV\_E\_NORESULT | 205 | No search results |
| NETDEV\_E\_SOCKET\_RECV\_ERROR | 206 | Socket failed to receive message |
| NETDEV\_E\_CREATE\_THREAD\_FAIL | 207 | Failed to create thread |
| NETDEV\_E\_RESCODE\_NO\_EXIST | 208 | Resource code does not exist |
| NETDEV\_E\_MSG\_DATA\_INVALID | 209 | Message content error |
| NETDEV\_E\_JSON\_NO\_IMAGE | 210 | No image |
| NETDEV\_E\_IMAGE\_SIZE\_BEYOND\_THE\_LIMIT | 211 | Image size exceeded the upper limit |
| NETDEV\_E\_MANAGETYPE\_NONSUPPORT | 403 | Cannot modify client/platform type |
| NETDEV\_E\_PUBLICKEYFAIL | 60068 | Failed to pair public key |

### Stream Media Related Error Code

|  |  |  |
| --- | --- | --- |
| **Code** | **Value** | **Meaning** |
| NETDEV\_E\_PLAYER\_FAIL | 1001 | Execution failure |
| NETDEV\_E\_PLAYER\_INVALID\_PARAM | 1002 | Invalid input parameter |
| NETDEV\_E\_PLAYER\_NO\_MEMORY | 1003 | Insufficient system memory |
| NETDEV\_E\_PLAYER\_SOCKET\_FAIL | 1004 | Failed to create socket |
| NETDEV\_E\_PLAYER\_RECV\_FAIL | 1005 | Failed to receive |
| NETDEV\_E\_PLAYER\_RECV\_ZERO | 1006 | Zero received |
| NETDEV\_E\_PLAYER\_NOT\_SUPPORT | 1007 | Unsupported function |
| NETDEV\_E\_PLAYER\_CREATETHREAD\_FAILED | 1008 | Failed to create thread |
| NETDEV\_E\_PLAYER\_OPENDL\_FAILED | 1009 | Failed to load dynamic library |
| NETDEV\_E\_PLAYER\_SYMDL\_FAILED | 1010 | Failed to get dynamic method |
| NETDEV\_E\_PLAYER\_SEND\_FAILED | 1011 | Failed to send |
| NETDEV\_E\_PLAYER\_EACCES | 1012 | No permission to create file |
| NETDEV\_E\_PLAYER\_FILE\_NOT\_FIND | 1013 | Failed to find the file to read |
| NETDEV\_E\_PLAYER\_LOG\_CLOSE | 1014 | Log closed |
| NETDEV\_E\_PLAYER\_MEDIA\_EXCEPTION | 1017 | Exception occurred during internal processing |
| NETDEV\_E\_PLAYER\_SYS\_FAIL | 1018 | Common system error |
| NETDEV\_E\_PLAYER\_INIT\_DONE | 1019 | Player is already initialized |
| NETDEV\_E\_PLAYER\_SYS\_RES\_FAILED | 1020 | Failed to create system resource |
| NETDEV\_E\_PLAYER\_INVALID\_IP | 1021 | IP error |
| NETDEV\_E\_PLAYER\_EZSTREAMER\_FULL | 1022 | Stream full |
| NETDEV\_E\_PLAYER\_VOD\_OVER\_ABILITY | 1023 | VMS stream full |
| NETDEV\_E\_PLAYER\_STREAM\_IN\_PROCESS | 1024 | Stream handling is in process |
| NETDEV\_E\_PLAYER\_NO\_SPARE\_SESSION | 1025 | No idle session |
| NETDEV\_E\_PLAYER\_NEED\_AUTHENTICATE | 1026 | Authentication required |
| NETDEV\_E\_PLAYER\_GET\_AUTHENTICATE\_FAID | 1027 | Authentication failed |
| NETDEV\_E\_PLAYER\_MAKE\_AUTHENTICATE\_FAID | 1028 | Failed to create authentication |
| NETDEV\_E\_PLAYER\_AUTHENTICATEINFO\_DIFF | 1029 | Inconsistent authentication information |
| NETDEV\_E\_PLAYER\_SESSION\_CLOSED | 1030 | Session closed |
| NETDEV\_E\_FAIL\_TO\_INIT\_EZPLAYER | 1257 | Failed to initialize player |
| NETDEV\_E\_FAIL\_TO\_ALLOC\_PORT\_RES | 1258 | Failed to allocate playing channel resource |
| NETDEV\_E\_FAIL\_TO\_GET\_PORT\_RES | 1259 | Failed to get playing channel resource |
| NETDEV\_E\_BUFFER\_QUEUE\_FULL | 1260 | Buffer queue is full |
| NETDEV\_E\_BUFFER\_QUEUE\_EMPTY | 1261 | Buffer queue is empty |
| NETDEV\_E\_OPEN\_FILE\_FAILED | 1262 | Failed to open file |
| NETDEV\_E\_FILE\_READ\_END | 1263 | File has been read |
| NETDEV\_E\_FILE\_DISKSPACE\_FULL | 1264 | Full disk space |
| NETDEV\_E\_FILE\_READ\_FAIL | 1265 | Failed to read file |
| NETDEV\_E\_MCM\_MIC\_NOT\_EXIST | 1266 | MIC does not exist |
| NETDEV\_E\_TS\_PACKET\_IN\_THE\_ROUGH | 1267 | TS packing not finished |
| NETDEV\_E\_FILE\_RECORD\_FINISH | 1268 | Recording has been saved |
| NETDEV\_E\_VIDEO\_RESOLUTION\_CHANGE | 1269 | Resolution changed |
| NETDEV\_E\_VIDEO\_RECORD\_PART | 1270 | Recording is segmented |
| NETDEV\_E\_FAIL\_TO\_OPEN\_STREAM | 1513 | Failed to start streaming |
| NETDEV\_E\_FAIL\_TO\_CLOSE\_STREAM | 1514 | Failed to stop streaming |
| NETDEV\_E\_FAIL\_TO\_RECV\_DATA | 1515 | Failed to receive data due to network fault |
| NETDEV\_E\_FAIL\_TO\_PROCESS\_MEDIA\_DATA | 1516 | Failed to process media data |
| NETDEV\_E\_NOT\_START\_PLAY | 1517 | Playing channel has not started playing operation |
| NETDEV\_E\_FAIL\_TO\_INPUT\_DATA | 1518 | Failed to input stream data |
| NETDEV\_E\_INPUTDATA\_BUFFER\_FULL | 1519 | Input data buffer is full |
| NETDEV\_E\_FAIL\_TO\_SET\_PROCESS\_DATA\_CB | 1520 | Failed to register the callback function to receive stream data |
| NETDEV\_E\_VOICE\_RUNNING | 1521 | Error occurred during audio service |
| NETDEV\_E\_FAIL\_TO\_OPEN\_VOICE\_SVC | 1522 | Failed to start audio service |
| NETDEV\_E\_FAIL\_TO\_CLOSE\_VOICE\_SVC | 1523 | Failed to stop audio service |
| NETDEV\_E\_UNKNOWN\_STREAM\_TYPE | 1524 | Unknown stream type |
| NETDEV\_E\_PACKET\_LOSE | 1525 | Packets lost |
| NETDEV\_E\_NEED\_MORE\_PACKET | 1526 | More packets are needed for packing |
| NETDEV\_E\_FAIL\_TO\_CREATE\_DECODE | 1527 | Failed to create decoder |
| NETDEV\_E\_FAIL\_TO\_DECODE | 1528 | Decoding failed |
| NETDEV\_E\_RECV\_DATA\_NOTENOUGH | 1529 | Data received is not enough |
| NETDEV\_E\_RENDER\_RES\_FULL | 1530 | Display resource is full |
| NETDEV\_E\_RENDER\_RES\_NOT\_EXIST | 1531 | Display resource does not exist |
| NETDEV\_E\_CREATE\_DEV\_FAILED | 1532 | Failed to create resource |
| NETDEV\_E\_AUDIO\_RES\_NOT\_EXIST | 1533 | Audio resource does not exist |
| NETDEV\_E\_IHW265D\_NEED\_MORE\_BITS | 1534 | More data required by decoder |
| NETDEV\_E\_FAIL\_TO\_CREATE\_ENCODE | 1535 | Failed to create encoder |
| NETDEV\_E\_CAPTURE\_RES\_EXIST | 1536 | Capture resource does not exist |
| NETDEV\_E\_RECORD\_STARTED | 1537 | Recording already started |
| NETDEV\_E\_NEED\_WAIT\_DECODEC | 1538 | Decoding not completed |
| NETDEV\_E\_MORE\_DATA\_NEED\_PACKET | 1539 | More data to be packed |
| NETDEV\_E\_AAC\_LC\_DECODE\_FAIL | 1540 | AAC\_LC decoding failed |
| NETDEV\_E\_RENDER\_SURFACELOST | 1541 | Render surface lost |
| NETDEV\_E\_PLAYBACK\_FINISH | 1542 | Playback ended |
| NETDEV\_E\_FILE\_ENCRYPED | 1543 | File is encrypted |
| NETDEV\_E\_SCRAMBLING\_INFO\_FAILED | 1544 | Scrambling info failed |
| NETDEV\_E\_LIVE\_EXISTED | 2000 | Live view already established |
| NETDEV\_E\_LIVE\_INPUT\_NOT\_READY | 2001 | Stream not ready |
| NETDEV\_E\_LIVE\_OUTPUT\_BUSY | 2002 | Live view display resource is busy |
| NETDEV\_E\_LIVE\_CB\_NOTEXIST | 2003 | Live view control block does not exist |
| NETDEV\_E\_LIVE\_STREAM\_FULL | 2004 | Live view stream resource is full |
| NETDEV\_E\_LIVE\_NET\_FAILED | 2005 | Session network error |
| NETDEV\_E\_LIVE\_NET\_TIMEOUT | 2006 | Session network timed out |
| NETDEV\_E\_LIVE\_SHAKE\_FAILED | 2007 | Session interaction error |
| NETDEV\_E\_LIVE\_AUTH\_FAILED | 2008 | Authentication failed |
| NETDEV\_E\_LIVE\_INNER\_ERROR | 2009 | Internal processing error on device side |
| NETDEV\_E\_LIVE\_INNER\_TIMEOUT | 2010 | Internal processing timed out |
| NETDEV\_E\_LIVE\_KEEP\_ALIVE\_FAILED | 2011 | Keep-alive failed |
| NETDEV\_E\_LIVE\_SESSION\_NOT\_EXIST | 2012 | Session does not exist |
| NETDEV\_E\_LIVE\_NOT\_ENOUGH\_BANDWIDTH2 | 2013 | Bandwidth not enough |
| NETDEV\_E\_LIVE\_REALPLAY\_ESTABLISHED | 2014 | Live view already established |
| NETDEV\_E\_LIVE\_REALPLAY\_RES\_BUSY | 2015 | Live view display resource is busy |
| NETDEV\_E\_LIVE\_MULTICAST\_DISABLED | 2016 | Multicast disabled |
| NETDEV\_E\_LIVE\_MULTICAST\_PORT\_OCCUPIED | 2017 | Multicast port has been occupied |
| NETDEV\_E\_LIVE\_MULTICAST\_PORT\_EXHAUSTED | 2018 | Multicast port has been exhausted |
| NETDEV\_E\_LIVE\_MULTICAST\_USER\_NOT\_EXIST | 2019 | Multicast user does not exist |
| NETDEV\_E\_LIVE\_CHANNEL\_NOT\_ONLINE | 2020 | Channel is offline |
| NETDEV\_E\_LIVE\_TALKBACK\_ENCODED\_INVALID | 2021 | Two-way audio resource code is invalid |
| NETDEV\_E\_LIVE\_VOICE\_RES\_USED\_BY\_TALKBACK | 2022 | Audio resource is being used by two-way audio |
| NETDEV\_E\_LIVE\_TALKBACK\_EXISTS | 2023 | Two-way audio already exists |
| NETDEV\_E\_LIVE\_VOICE\_WORK\_NOT\_EXIST | 2024 | Two-way audio does not exist |
| NETDEV\_E\_LIVE\_TALKBACK\_TIMEOUT | 2025 | Two-way audio timed out |
| NETDEV\_E\_LIVE\_TALKBACK\_ERROR | 2026 | Two-way audio failed |
| NETDEV\_E\_LIVE\_UNDEFINED\_ERROR | 2027 | Undefined error |
| NETDEV\_E\_LIVE\_BAD\_REQUEST | 2028 | Bad request |
| NETDEV\_E\_LIVE\_UNAUTHORIZED | 2029 | Unauthorized |
| NETDEV\_E\_LIVE\_PAYMENT\_REQUIRED | 2030 | Payment required |
| NETDEV\_E\_LIVE\_FORIBIDDEN | 2031 | Forbidden |
| NETDEV\_E\_LIVE\_METHOD\_NOT\_ALLOWED | 2032 | Method not allowed |
| NETDEV\_E\_LIVE\_NOT\_ACCEPTABLE | 2033 | Not acceptable |
| NETDEV\_E\_LIVE\_PROXY\_REQUIRED | 2034 | Proxy requires authentication |
| NETDEV\_E\_LIVE\_REQUEST\_TIMEOUT | 2035 | Request timed out |
| NETDEV\_E\_LIVE\_GONE | 2036 | Not on server |
| NETDEV\_E\_LIVE\_LENGTH\_REQUIRED | 2037 | Length required |
| NETDEV\_E\_LIVE\_PRECONDITION\_FAILED | 2038 | Pre-processing failed |
| NETDEV\_E\_LIVE\_ENTITY\_TOO\_LARGE | 2039 | Entity too long |
| NETDEV\_E\_LIVE\_URI\_TOO\_LARGE | 2040 | URI too long |
| NETDEV\_E\_LIVE\_UNSUPPORTED\_TYPE | 2041 | Unsupported media type |
| NETDEV\_E\_LIVE\_NOT\_UNDERSTOOD | 2042 | Parameter not understood |
| NETDEV\_E\_LIVE\_CONFERENCE\_NOT\_FOUND | 2043 | Conference not found |
| NETDEV\_E\_LIVE\_NOT\_ENOUGH\_BANDWIDTH | 2044 | Bandwidth not enough |
| NETDEV\_E\_LIVE\_SESSION\_NOT\_FOUND | 2045 | Session not found |
| NETDEV\_E\_LIVE\_METHOD\_NOT\_VALID | 2046 | Invalid method in this state |
| NETDEV\_E\_LIVE\_HEADER\_NOT\_VALID | 2047 | Invalid header for the resource |
| NETDEV\_E\_LIVE\_INVALID\_RANGE | 2048 | Invalid range |
| NETDEV\_E\_LIVE\_PARAMETER\_READ\_ONLY | 2049 | Parameter is read only |
| NETDEV\_E\_LIVE\_AO\_NOT\_ALLOWED | 2050 | AO is not allowed |
| NETDEV\_E\_LIVE\_ONLY\_AO\_ALLOWED | 2051 | Only AO is allowed |
| NETDEV\_E\_LIVE\_UNSUPPORTED\_TRANSPORT | 2052 | Unsupported transport mode |
| NETDEV\_E\_LIVE\_DESTINATION\_UNREACHABLE | 2053 | Destination is unreachable |
| NETDEV\_E\_LIVE\_INTERNAL\_SERVER\_ERROR | 2054 | Server internal error |
| NETDEV\_E\_LIVE\_NOT\_IMPLEMENTED | 2055 | Not implemented |
| NETDEV\_E\_LIVE\_BAD\_GATEWAY | 2056 | Gateway error |
| NETDEV\_E\_LIVE\_SERVICE\_UNAVAILABLE | 2057 | Service unavailable |
| NETDEV\_E\_LIVE\_VERSION\_NOT\_SUPPORTED | 2058 | Unsupported RTSP version |
| NETDEV\_E\_LIVE\_GATEWAY\_TIMEOUT | 2059 | Gateway timed out |
| NETDEV\_E\_LIVE\_OPTION\_NOT\_SUPPORTED | 2060 | Unsupported option |
| NETDEV\_E\_LIVE\_MALLOC\_FAIL | 2061 | Failed to allocate memory |
| NETDEV\_E\_LIVE\_REALLOC\_FAIL | 2062 | Failed to re-allocate memory |
| NETDEV\_E\_LIVE\_DESCRIBE\_TIMEOUT | 2063 | Describe timed out |
| NETDEV\_E\_LIVE\_IPC\_NOTBIND | 2064 | Channel is not bound |
| NETDEV\_E\_LIVE\_DISK\_ABNOMAL | 2065 | Disk abnormal |
| NETDEV\_E\_AUDIO\_EXISTED | 2100 | Two-way audio already exists |
| NETDEV\_E\_AUDIO\_NO\_EXISTED | 2101 | Two-way audio does not exist |
| NETDEV\_E\_AUDIO\_RESCODE\_INVALID | 2102 | Two-way audio resource code is invalid |
| NETDEV\_E\_AUDIO\_FAILED | 2104 | Two-way audio failed |
| NETDEV\_E\_AUDIO\_AUDIOBCAST\_FULL | 2205 | Audio service is full |
| NETDEV\_E\_CAPTURE\_NO\_SUPPORT\_FORMAT | 2200 | Unsupported snapshot format |
| NETDEV\_E\_CAPTURE\_NO\_ENOUGH\_CAPACITY | 2201 | Insufficient disk space |
| NETDEV\_E\_CAPTURE\_NO\_DECODED\_PICTURE | 2202 | No decoded image available for snapshot |
| NETDEV\_E\_CAPTURE\_SINGLE\_FAILED | 2203 | Single snapshot failed |
| NETDEV\_E\_OVER\_ABILITY | 2301 | Stream is beyond the capability |
| NETDEV\_E\_CLOUD\_DOWNLOAD\_FINISH | 2793 | Download finished |
| NETDEV\_E\_CLOUD\_PARSE\_DOMAIN\_FAIL | 2794 | Failed to resolve domain name |
| NETDEV\_E\_CLOUD\_CONNECT\_FAIL | 2795 | Connection failed |
| NETDEV\_E\_CLOUD\_CONNECT\_TIMEOUT | 2796 | Connection timed out |
| NETDEV\_E\_CLOUD\_DOWNLOAD\_TIMEOUT | 2797 | Download timed out |
| NETDEV\_E\_CLOUD\_DOWNLOAD\_FAIL | 2798 | Download failed |
| NETDEV\_E\_CLOUD\_NETWORK\_POOR | 2799 | Poor network |
| NETDEV\_E\_CLOUD\_PLAY\_FINISH | 2800 | Playing finished |
| NETDEV\_E\_CLOUD\_DISK\_FULL | 2801 | Full disk space |
| NETDEV\_E\_CLOUD\_AUTH\_FAIL | 2802 | Authentication failed |
| NETDEV\_E\_CLOUD\_CURRENT\_TIME | 2803 | Current play time (for reporting use only) |
| NETDEV\_E\_CLOUD\_PRIOR\_DISK\_FULL | 2804 | Used disk space reached the preset value |
| NETDEV\_E\_CLOUD\_NODE\_NOT\_EXIST | 2805 | Time node does not exist |
| NETDEV\_E\_CLOUD\_NO\_CACHE\_PATH | 2806 | No cache path |
| NETDEV\_E\_CLOUD\_MSG\_SEND\_FAIL | 2807 | Failed to send message |
| NETDEV\_E\_CLOUD\_TASK\_CANCELLED | 2808 | Task has been canceled |
| NETDEV\_E\_CLOUD\_TASK\_STREAM\_CONTINUE | 2809 | Stream continues to play |
| NETDEV\_E\_MEDIA\_INPUT\_NOT\_READY | 10000 | Stream not ready |
| NETDEV\_E\_CCB\_STATR\_INVALID | 10001 | Control block status is invalid |
| NETDEV\_E\_MEDIA\_OUTPUT\_BUSY | 10002 | Live view display resource is busy |
| NETDEV\_E\_MEDIA\_START\_LOCAL\_LIVE\_ERR | 10003 | Live view stream not ready |
| NETDEV\_E\_MEDIA\_START\_LOCAL\_REPLAY\_ERR | 10004 | Playback stream not ready |
| NETDEV\_E\_MEDIA\_BW\_RECV\_NOT\_ENOUGH | 10007 | Insufficient network bandwidth for receiving data |
| NETDEV\_E\_MEDIA\_BW\_SEND\_NOT\_ENOUGH | 10008 | Insufficient network bandwidth for sending data |
| NETDEV\_E\_MEDIA\_AUDIO\_BROADCAST\_TO\_LIMIT | 10009 | Broadcast service has reached the upper limit |
| NETDEV\_E\_MEDIA\_AUDIO\_CHL\_BING\_USED | 10010 | Audio channel is being used |
| NETDEV\_E\_MEDIA\_NOT\_SUPPORT\_ENCODETYPE | 10012 | Unsupported stream format |
| NETDEV\_E\_MEDIA\_MAX | 10399 | Max media related error code |

### Error Codes of PTZ Module

|  |  |  |
| --- | --- | --- |
| **Error Code** | **Value** | **Meaning** |
| NETDEV\_E\_SET\_PRESET\_FAILED | 10400 | Failed to set preset |
| NETDEV\_E\_QUERY\_PRESET\_FAILED | 10401 | Failed to query preset |
| NETDEV\_E\_QUERY\_TRACK\_FAILED | 10402 | Failed to query route |
| NETDEV\_E\_START\_RECORD\_TRACK\_FAILED | 10403 | Failed to start recording route |
| NETDEV\_E\_STOP\_RECORD\_TRACK\_FAILED | 10404 | Failed to stop recording route |
| NETDEV\_E\_QUERY\_CRUISE\_FAILED | 10405 | Failed to query patrol route |
| NETDEV\_E\_SET\_CRUISE\_FAILED | 10406 | Failed to set patrol route |
| NETDEV\_E\_PTZ\_COMMAND\_FAILED | 10407 | PTZ operation failed |
| NETDEV\_E\_PRESET\_IN\_CRUISE | 10408 | Preset is being used by patrol route and cannot be deleted |
| NETDEV\_E\_CRUISEPOINT\_ER | 10409 | Discontinuous patrol points |
| NETDEV\_E\_TRACK\_ISUSED | 10410 | Route being used and cannot be deleted |
| NETDEV\_E\_SERIALMODE\_MISMATCH | 10411 | Serial port mode not match |
| NETDEV\_E\_TRACK\_NOT\_EXIST | 10412 | Route does not exist |
| NETDEV\_E\_MODE\_CRUISE\_FULL | 10413 | Route points are full |
| NETDEV\_E\_PTZ\_ISUSED | 10414 | PTZ is in use |
| NETDEV\_E\_PRESET\_IN\_GUARD | 10415 | Preset is being used by auto guard |
| NETDEV\_E\_CRUISE\_PATH\_IN\_GUARD | 10416 | Patrol route is being used by auto guard |
| NETDEV\_E\_PTZ\_MAX | 10799 | Max error code of PTZ module |

### Error Code of Playback Module

|  |  |  |
| --- | --- | --- |
| **Code** | **Value** | **Meaning** |
| NETDEV\_E\_VOD\_ABEND | 10800 | Playback ended abnormally |
| NETDEV\_E\_VOD\_END | 10801 | Playback ended |
| NETDEV\_E\_VOD\_CALL\_DRV | 10802 | Failed to call driver interface |
| NETDEV\_E\_VOD\_CB\_NOT\_FIND | 10803 | Control block was not found |
| NETDEV\_E\_VOD\_OVER\_ABILITY | 10804 | Playback capability exceeded |
| NETDEV\_E\_VOD\_TAG\_OVER\_LIMIT | 10805 | Number of tags exceeds the upper limit |
| NETDEV\_E\_VOD\_NO\_RECORDING | 10806 | No recording content |
| NETDEV\_E\_VOD\_SMART\_NOT\_SUPPORT | 10807 | GB28181 device does not support smart playback |
| NETDEV\_E\_VOD\_NO\_REPLAYURL | 10808 | Cannot get playback URL |
| NETDEV\_E\_VOD\_MAX | 10999 | Max error code of playback module |

### Error Code of Storage Module

|  |  |  |
| --- | --- | --- |
| **Code** | **Value** | **Meaning** |
| NETDEV\_E\_STOR\_ABEND | 101000 | Storage ended abnormally |
| NETDEV\_E\_ARRAY\_NAME\_EXITS | 11001 | Array already exists |
| NETDEV\_E\_NOT\_RAID\_MODE | 11002 | Not RAID mode |
| NETDEV\_E\_NO\_STORE\_PLAN | 11003 | Recording schedule is not configured for the channel |
| NETDEV\_E\_DISK\_ZERO\_OR\_ABNORMAL | 11022 | No disk or disk is abnormal |
| NETDEV\_E\_SLOT\_NOT\_NORMAL | 11023 | Slot 1 disk is abnormal |
| NETDEV\_E\_SLOT\_RECONNECT | 11024 | Restart is required if slot 1 disk is pulled out and then re-plugged in after face recognition is enabled |
| NETDEV\_E\_BATTERY\_STATUS\_FAULT | 11034 | Device battery status is abnormal |
| NETDEV\_E\_STOR\_MAX | 11199 | Max error code of storage module |

### Error Code of User Management Module

|  |  |  |
| --- | --- | --- |
| **Code** | **Value** | **Meaning** |
| NETDEV\_E\_USER\_NOT\_ONLINE | 101200 | User is offline |
| NETDEV\_E\_USER\_WRONG\_PASSWD | 11201 | Incorrect password |
| NETDEV\_E\_USER\_NO\_SUCH\_USER | 11202 | User does not exist |
| NETDEV\_E\_USER\_NO\_AUTH | 11203 | User has no permission |
| NETDEV\_E\_USER\_MAX\_NUM | 11204 | Number of users added reached the upper limit |
| NETDEV\_E\_USER\_EXIST | 11205 | User already exists |
| NETDEV\_E\_USER\_LOGIN\_MAX\_NUM | 11206 | Number of logged-in users reached upper limit |
| NETDEV\_E\_USER\_LOCKED | 11207 | User has been locked |
| NETDEV\_E\_USER\_IS\_ADMIN | 11208 | The user is administrator and the operation is not allowed |
| NETDEV\_E\_USER\_NOT\_VALID\_PERIOD | 11209 | Invalid period |
| NETDEV\_E\_USER\_ROLE\_BEYOND\_RANGE | 11210 | Number of user roles reached upper limit |
| NETDEV\_E\_USER\_SELOG\_IP\_LOCKED | 11211 | User is locked for security |
| NETDEV\_E\_ROLE\_EXIST | 11212 | Role already exists |
| NETDEV\_E\_USER\_NOT\_VALID\_TIME | 11213 | Invalid period |
| NETDEV\_E\_USER\_WEAK\_PASSWD\_LOGIN\_FAIL | 11214 | Login failed. Please change the password into a strong one locally and try again. |
| NETDEV\_E\_USER\_IS\_TEMPPASSWD | 11217 | User is using temporary password |
| NETDEV\_E\_USER\_TEMPPASSWD\_LOGIN\_FAIL | 11218 | Temporary password is forbidden on WAN |
| NETDEV\_E\_USER\_WEAK\_PASSWD\_MODIFY | 11219 | Weak password is forbidden |
| NETDEV\_E\_USER\_IP\_NO\_AUTH | 11221 | Login IP has no authorization |
| NETDEV\_E\_USER\_MAX | 11599 | Max error code of user module |

### Error Code of Device Management Module

|  |  |  |
| --- | --- | --- |
| **Code** | **Value** | **Meaning** |
| NETDEV\_E\_DEV\_REPEAT\_DEV | 11602 | Duplicate device |
| NETDEV\_E\_DEV\_NO\_SUCH\_DEV | 11603 | Device does not exist |
| NETDEV\_E\_DEV\_NO\_SUCH\_CHL | 11604 | Channel does not exist |
| NETDEV\_E\_DEV\_MAX\_NUM | 11605 | Number of devices added to VMS reached the upper limit |
| NETDEV\_E\_DEV\_LOGIN\_MAX\_NUM | 11606 | Number of logins to device reached the upper limit |
| NETDEV\_E\_DEV\_COMMON\_FAILED | 11607 | Common error returned by device |
| NETDEV\_E\_DEV\_NO\_SUCH\_DEV\_OID | 11608 | Old device ID does not exist |
| NETDEV\_E\_DEV\_DEL\_FORBID | 11609 | Cannot delete device |
| NETDEV\_E\_CHL\_OFFLINE | 11610 | Channel offline |
| NETDEV\_E\_DEV\_TYPE\_UNMATCHED | 11611 | Device type not match |
| NETDEV\_E\_DEV\_REPEAT\_DEV\_NAME | 11612 | Duplicate device name |
| NETDEV\_E\_DEV\_CFG\_FAILED | 11613 | Failed to configure device |
| NETDEV\_E\_DEV\_NO\_RESULT | 11614 | No search results |
| NETDEV\_E\_DEV\_DEC\_OFFLINE | 11617 | Decoding device offline |
| NETDEV\_E\_DEV\_VCA\_ALARM\_NO\_PIC | 11661 | Failed to get VCA alarm image |
| NETDEV\_E\_COMMON\_FAIL\_STAT | 11801 | Failed to read file (directory) status |
| NETDEV\_E\_COMMON\_FILE\_NONEXIST | 11802 | File does not exist |
| NETDEV\_E\_COMMON\_MKDIR\_ERR | 11803 | Failed to create directory |
| NETDEV\_E\_SUBSCRIBE\_FULL | 11804 | Subscription is full for current user (cannot return error code when subscription is full at device side) |
| NETDEV\_E\_UPGRADE\_NOTADMIN | 11805 | Only admin can upgrade |
| NETDEV\_E\_UPGRADE\_INVALID | 11806 | Upgrade not started |
| NETDEV\_E\_UPGRADE\_INPROCESS | 11807 | Upgrade in progress |
| NETDEV\_E\_UPGRADE\_NOMEMORY | 11808 | Insufficient memory for upgrade |
| NETDEV\_E\_UPGRADE\_FILE\_OPEN\_ERR | 11809 | Error occurred while opening the mirror file during upgrade |
| NETDEV\_E\_UPGRADE\_DEVICE\_ERR | 11810 | Error occurred while upgrading flash |
| NETDEV\_E\_UPGRADE\_BUSY | 11811 | Cannot load multiple upgrade processes simultaneously |
| NETDEV\_E\_UPGRADE\_FAIL\_TIMEOUT | 11812 | Upgrade timed out |
| NETDEV\_E\_INVALID\_CONFIGFILE | 11813 | Invalid configuration file |
| NETDEV\_E\_STOR\_RESOURCE\_NOTINIT | 11814 | Storage resource not allocated |
| NETDEV\_E\_DEV\_MAX | 11999 | Max error code of device module |

### Error Code of Organization Management Module

|  |  |  |
| --- | --- | --- |
| **Code** | **Value** | **Meaning** |
| NETDEV\_E\_ORG\_MAX\_NUM | 12000 | Number of organizations reached the upper limit |
| NETDEV\_E\_ORG\_MAX\_LEVEL | 12001 | Number of organization levels reached the upper limit |
| NETDEV\_E\_ORG\_RESOURCE\_EXIST | 12002 | Cannot delete an organization containing resources |
| NETDEV\_E\_ORG\_EXIST | 12003 | Organization already exists |
| NETDEV\_E\_ORG\_NO\_SUCH\_ORG | 12004 | Organization does not exist |
| NETDEV\_E\_ORG\_IS\_ROOT | 12005 | Cannot delete root organization |
| NETDEV\_E\_ORG\_NO\_SUCH\_PORG | 12006 | Parent organization does not exist |
| NETDEV\_E\_ORG\_SUB\_EXIST | 12007 | Cannot delete an organization containing sub organizations |
| NETDEV\_E\_ORG\_MAX | 12399 | Max error code of organization module |

### Error Code of Video Wall Management Module

|  |  |  |
| --- | --- | --- |
| **Code** | **Value** | **Meaning** |
| NETDEV\_E\_TVWALL\_WND\_NUMLIMITED | 12400 | Number of windows exceeds upper limit |
| NETDEV\_E\_TVWALL\_NUMLIMTED | 12401 | Number of video walls exceeded upper limit |
| NETDEV\_E\_TVWALL\_LED\_NUMLIMTED | 12402 | Number of virtual LEDs exceeds upper limit |
| NETDEV\_E\_TVWALL\_LED\_LINELIMTED | 12403 | Number of lines in virtual LED exceeds upper limit |
| NETDEV\_E\_TVWALL\_SCENE\_OVER\_LIMIT | 12404 | Number of scenes on video wall exceeds upper limit |
| NETDEV\_E\_TVWALL\_BASEPIC\_NUMLIMITED | 12405 | Number of background images that can be saved exceeds upper limit |
| NETDEV\_E\_TVWALL\_BASEPIC\_SCENELIMITED | 12406 | Number of background images that can be bound to a scene exceeds upper limit |
| NETDEV\_E\_TVWALL\_DECODE\_OUTRANGE | 12407 | Target position exceeds decoding capability |
| NETDEV\_E\_TVWALL\_WND\_OUTRANGE | 12408 | Window size exceeds upper limit |
| NETDEV\_E\_TVWALL\_WND\_RANGELIMITED | 12409 | Window/roaming position out of range |
| NETDEV\_E\_TVWALL\_SCROLL\_LED\_NUMLIMITED | 12410 | Number of scrolling virtual LEDs exceeds upper limit |
| NETDEV\_E\_TVWALL\_WND\_LOCKED | 12430 | Current window is locked |
| NETDEV\_E\_TVWALL\_LED\_NOTEXIST | 12431 | Virtual LED does not exist |
| NETDEV\_E\_TVWALL\_SEQ\_EXISTD | 12432 | Sequence already exists |
| NETDEV\_E\_TVWALL\_WND\_NOTEXIST | 12433 | Window does not exist |
| NETDEV\_E\_TVWALL\_WND\_EXIST | 12434 | Window already exists |
| NETDEV\_E\_TVWALL\_SPILT\_NOTEXIST | 12435 | Split window does not exist |
| NETDEV\_E\_TVWALL\_WND\_NOSPILT | 12436 | Window cannot split |
| NETDEV\_E\_TVWALL\_CHANNEL\_UESED | 12437 | Decoding channel is in use by other video wall |
| NETDEV\_E\_TVWALL\_WINDEXIST | 12438 | Window exists on video wall |
| NETDEV\_E\_TVWALL\_NOTEXIST | 12439 | Video wall info does not exist |
| NETDEV\_E\_TVWALL\_SPLIT\_INFOERR | 12440 | Incorrect split type |
| NETDEV\_E\_TVWALL\_CHANNEL\_NUMERR | 12411 | Incorrect channel number |
| NETDEV\_E\_TVWALL\_ENC\_ERR | 12442 | Stream not ready |
| NETDEV\_E\_TVWALL\_PERSCREEN\_NUMLIMITED | 12443 | Number of windows per VO exceeds upper limit |
| NETDEV\_E\_TVWALL\_MAX | 12799 | Max error code of video wall module |

### Network Related Error Code

|  |  |  |
| --- | --- | --- |
| **Code** | **Value** | **Meaning** |
| NETDEV\_E\_IPADDR\_CONFLICT | 12800 | IP address conflict |
| NETDEV\_E\_UPNP\_PORT\_DISCARD | 12801 | Mapping port has been discarded |
| NETDEV\_E\_UPNP\_PORT\_EMPLOY | 12802 | Port is already in use |
| NETDEV\_E\_SET\_VMP\_PORT\_FAILED | 12803 | Failed to set VMP port |
| NETDEV\_E\_SET\_MEDIA\_PORT\_FAILED | 12804 | Failed to set media port |
| NETDEV\_E\_SET\_RTSP\_PORT\_FAILED | 12805 | Failed to set RTSP port |
| NETDEV\_E\_SET\_ONVIF\_PORT\_FAILED | 12806 | Failed to set ONVIF port |
| NETDEV\_E\_SET\_HTTPS\_PORT\_FAILED | 12807 | Failed to set HTTPS port |
| NETDEV\_E\_SET\_HTTP\_PORT\_FAILED | 12808 | Failed to set HTTP port |
| NETDEV\_E\_TEST\_EMAIL | 12809 | Failed to send test email |
| NETDEV\_E\_DDNS\_INIT | 12810 | Initialize defaults |
| NETDEV\_E\_DDNS\_GOOD | 12811 | Update completed |
| NETDEV\_E\_DDNS\_NOCHG | 12812 | Update completed but IP does not change |
| NETDEV\_E\_DDNS\_BADAUTH | 12813 | Incorrect user password or password |
| NETDEV\_E\_DDNS\_DONATOR | 12814 | Account-related error. An option available only to credited users (such as offline URL) |
| NETDEV\_E\_DDNS\_NOTFQ | 12815 | Invalid host name format |
| NETDEV\_E\_DDNS\_NOHOST | 12816 | Host name account does not exist |
| NETDEV\_E\_DDNS\_NUMHOST | 12817 | Request immediate update for more than 20 hosts |
| NETDEV\_E\_DDNS\_ABUSE | 12818 | Host name is banned for overuse. |
| NETDEV\_E\_DDNS\_BADAGENT | 12819 | Invalid format of HTTP request sent by agent |
| NETDEV\_E\_DDNS\_GOOD\_127 | 12820 | Indicates error in HTTP request specification unless the client wants to update 127.0.0.1 |
| NETDEV\_E\_DDNS\_DNSERR | 12821 | DNS error. Please contact support |
| NETDEV\_E\_DDNS\_911 | 12822 | Being processed by service maintenance |
| NETDEV\_E\_DDNS\_REQ | 12823 | Failed to send DDNS request |
| NETDEV\_E\_DDNS\_CHECK\_KEY | 12824 | Incorrect verification code |
| NETDEV\_E\_DDNS\_BAD\_NAME\_FORMAT | 12825 | Invalid domain name format |
| NETDEV\_E\_DDNS\_DOMAIN\_CONFLICT | 12826 | Conflicting domain name |
| NETDEV\_E\_DDNS\_CLOSE | 12827 | Not returned by DDNS server, only used to close DDNS record status |
| NETDEV\_E\_CREATE\_CERT\_FAILD | 12828 | Failed to generate certificate file |
| NETDEV\_E\_GET\_CERT\_FAILD | 12829 | Uploaded certificate file is invalid |
| NETDEV\_E\_TEST\_EMAIL\_CHECKSERVER\_INFO | 12830 | Incorrect username or password for authentication server |
| NETDEV\_E\_TEST\_EMAIL\_COMM | 12831 | Failed to send mail. Please check network and email configurations. |
| NETDEV\_E\_TEST\_EMAIL\_TIMEOUT | 12832 | Email sending timeout |
| NETDEV\_E\_CLOUD\_REPEAT\_USER | 12833 | The cloud account is already logged in |
| NETDEV\_E\_CLOUD\_DEV\_ADD\_FAIL | 12834 | Failed to add cloud device info |
| NETDEV\_E\_CLOUD\_ORG\_NULL | 12835 | Empty cloud organization info |
| NETDEV\_E\_GET\_CLOUD\_ORG | 12836 | Failed to get cloud info. Please check network configurations |
| NETDEV\_E\_NO\_SUCH\_CLOUD\_USER | 12837 | Cloud account does not exist |
| NETDEV\_E\_NET\_MAX | 13199 | Max error code of network module |

# Definition of Product Related Interfaces

## Common Interfaces

|  |  |
| --- | --- |
| **Function** | **Interface** |
| SDK initialization | NETDEV\_Init |
| SDK cleanup | NETDEV\_Cleanup |
| Set log size and quantity | NETDEV\_ConfigLogFile |
| Set log path | NETDEV\_SetLogPath |
| Set the path to write to logs | NETDEV\_SetWriteLogFlag |
| Get SDK version | NETDEV\_GetSDKVersion |
| Get interface error code | NETDEV\_GetLastError |
| Register the callback function to receive exceptions | NETDEV\_SetExceptionCallBack |
| Set the maximum number of decoding channels | NETDEV\_SetMaxDecChlNum |
| Set keep-alive interval and times | NETDEV\_SetConnectTime |
| Set receiving timeout | NETDEV\_SetRevTimeOut |
| Register the callback function to discover devices | NETDEV\_SetDiscoveryCallBack |
| Search device | NETDEV\_Discovery |
| Log in to device | NETDEV\_Login\_V30 |
| Log out of device | NETDEV\_Logout |
| Generate a keyframe dynamically | NETDEV\_MakeKeyFrame |
| Start live view | NETDEV\_RealPlay |
| Stop live view | NETDEV\_StopRealPlay |
| Start local recording | NETDEV\_SaveRealData |
| Stop local recording | NETDEV\_StopSaveRealData |
| Get live video URL | NETDEV\_GetStreamUrl |
| Start live video by URL | NETDEV\_FastRealPlayByUrl |
| Search recording file list | NETDEV\_FindFile |
| Get file info one by one | NETDEV\_FindNextFile |
| Close recording file search | NETDEV\_FindClose |
| LAPI alarm subscription | NETDEV\_SubscibeLapiAlarm |
| Cancel LAPI alarm subscription | NETDEV\_UnSubLapiAlarm |
| Register the callback function to receive raw streams | NETDEV\_SetPlayDataCallBack |
| Register the callback function to receive assembled live stream data | NETDEV\_SetPlayParseCB |
| Register the callback function to receive the decoded video stream data | NETDEV\_SetPlayDecodeVideoCB |
| Register the callback function to receive the assembled audio stream data | NETDEV\_SetPlayParseAudioCB |
| Register the callback function to receive the decoded audio stream data | NETDEV\_SetPlayDecodeAudioCB |
| Register the callback function to receive the displayed data | NETDEV\_SetPlayDisplayCB |
| Get bit rate of window | NETDEV\_GetBitRate |
| Get frame rate of window | NETDEV\_GetFrameRate |
| Get encoding format of window. | NETDEV\_GetVideoEncodeFmt |
| Get video resolution. | NETDEV\_GetResolution |
| Get packet loss rate of window. | NETDEV\_GetLostPacketRate |
| Reset packet loss rate of window | NETDEV\_ResetLostPacketRate |
| Set image display scale | NETDEV\_SetRenderScale |
| Get image parameters (current image parameters only) | NETDEV\_GetVideoEffect |
| Adjust current image | NETDEV\_SetVideoEffect |
| Set digital zoom | NETDEV\_SetDigitalZoom |
| Live view snapshot | NETDEV\_CapturePicture |
| Set fluency priority | NETDEV\_SetPictureFluency |
| Start two-way audio | NETDEV\_StartVoiceCom |
| Stop two-way audio | NETDEV\_StopVoiceCom |
| Turn on microphone | NETDEV\_OpenMic |
| Turn off microphone | NETDEV\_CloseMic |
| Adjust microphone volume | NETDEV\_MicVolumeControl |
| Get microphone volume | NETDEV\_GetMicVolume |
| Get mute status | NETDEV\_GetMuteStatus |
| Set mute status | NETDEV\_SetMuteStatus |
| Turn on sound | NETDEV\_OpenSound |
| Adjust speaker volume | NETDEV\_SoundVolumeControl |
| Get speaker volume | NETDEV\_GetSoundVolume |
| Turn off sound | NETDEV\_CloseSound |
| Star audio data input service | NETDEV\_StartInputVoiceSrv |
| Stop audio data input service | NETDEV\_StopInputVoiceSrv |
| Input audio data | NETDEV\_InputVoiceData |
| Register the callback function to receive parsed audio data | NETDEV\_SetParseVoiceDataCB |
| Play recording file by time | NETDEV\_PlayBackByTime |
| Control playback status | NETDEV\_PlayBackControl |
| Stop playback | NETDEV\_StopPlayBack |
| Download recording file by time | NETDEV\_GetFileByTime |
| Stop downloading recording file | NETDEV\_StopGetFile |
| Open local file | NETDEV\_OpenMediaFile |
| Play local file | NETDEV\_StartPlayMediaFile |
| Stop playing local file | NETDEV\_StopPlayMediaFile |
| Get total duration of media file | NETDEV\_GetMediaFileTime |
| Perform PTZ control (live view not started) | NETDEV\_PTZControl\_Other |
| Get preset position list | NETDEV\_GetPTZPresetList |
| Operate presets (no need to start preview) | NETDEV\_PTZPreset\_Other |
| Get preset patrol route | NETDEV\_PTZGetCruise |
| Operate preset patrol (no need to start preview) | NETDEV\_PTZCruise\_Other |
| Get recorded patrol route | NETDEV\_PTZGetTrackCruise |
| Operate recorded patrol (no need to start preview) | NETDEV\_PTZTrackCruise |
| Drag to zoom in/out (no need to start preview) | NETDEV\_PTZSelZoomIn\_Other |
| 3D positioning (no need to start preview) | NETDEV\_PTZ3DPosition\_Other |
| Add person library info | NETDEV\_CreatePersonLibInfo |
| Modify person library info | NETDEV\_ModifyPersonLibInfo |
| Search all the created person library information. | NETDEV\_FindPersonLibList |
| Get person library info one by one | NETDEV\_FindNextPersonLibInfo |
| Stop searching face library and release resource | NETDEV\_FindClosePersonLibList |
| Delete specified person library | NETDEV\_DeletePersonLibInfo |
| Add specified person info | NETDEV\_AddPersonInfo |
| Modify specified person info | NETDEV\_ModifyPersonInfo |
| Delete specified person info | NETDEV\_DeletePersonInfo |
| Search all face monitoring tasks | NETDEV\_FindPersonMonitorList |
| Get monitoring task info one by one | NETDEV\_FindNextPersonMonitorInfo |
| Stop searching monitoring task and release resource | NETDEV\_FindClosePersonMonitorList |
| Add a single face monitoring task | NETDEV\_AddPersonMonitorInfo |
| Delete a single face monitoring task | NETDEV\_DeletePersonMonitorInfo |
| Get info about a single face monitoring task | NETDEV\_GetPersonMonitorRuleInfo |
| Set a single face monitoring task | NETDEV\_SetPersonMonitorRuleInfo |
| Delete face monitoring tasks in batches | NETDEV\_BatchDeletePersonMonitorInfo |
| Ge capacity info of all person libraries | NETDEV\_GetPersonLibCapacity |
| Search person info with criteria | NETDEV\_FindPersonInfoList |
| Get person info one by one | NETDEV\_FindNextPersonInfo |
| Stop searching person info and release resource | NETDEV\_FindClosePersonInfoList |
| Register the callback function to receive face recognition alarm | NETDEV\_SetPersonAlarmCallBack |
| Register the callback function to receive structured alarm | NETDEV\_SetStructAlarmCallBack |

## Network Camera

|  |  |
| --- | --- |
| **Function** | **Interface** |
| Get video channel list | NETDEV\_QueryVideoChlDetailList |
| Get video channel list | NETDEV\_QueryVideoChlDetailListEx |
| Start live view | NETDEV\_RealPlay |
| Stop live view | NETDEV\_StopRealPlay |
| Register the callback function to receive raw streams | NETDEV\_SetPlayDataCallBack |
| Register the callback function to receive assembled live stream data | NETDEV\_SetPlayParseCB |
| Register the callback function to receive the decoded video stream data | NETDEV\_SetPlayDecodeVideoCB |
| Register the callback function to receive the assembled audio stream data | NETDEV\_SetPlayParseAudioCB |
| Register the callback function to receive the decoded audio stream data | NETDEV\_SetPlayDecodeAudioCB |
| Register the stream callback function to set the callback of displayed data | NETDEV\_SetPlayDisplayCB |
| Get bit rate of window | NETDEV\_GetBitRate |
| Get frame rate of window | NETDEV\_GetFrameRate |
| Get encoding format of window | NETDEV\_GetVideoEncodeFmt |
| Get video resolution | NETDEV\_GetResolution |
| Get packet loss rate of window | NETDEV\_GetLostPacketRate |
| Reset packet loss rate of window | NETDEV\_ResetLostPacketRate |
| Set image display scale | NETDEV\_SetRenderScale |
| Get image parameters (current image parameters only) | NETDEV\_GetVideoEffect |
| Adjust current image | NETDEV\_SetVideoEffect |
| Set digital zoom | NETDEV\_SetDigitalZoom |
| Live view snapshot | NETDEV\_CapturePicture |
| Snapshot (not in preview) | NETDEV\_CaptureNoPreview |
| Generate a keyframe dynamically | NETDEV\_MakeKeyFrame |
| Local recording | NETDEV\_SaveRealData |
| Stop local recording | NETDEV\_StopSaveRealData |
| Set fluency priority | NETDEV\_SetPictureFluency |
| Start two-way audio | NETDEV\_StartVoiceCom |
| Stop two-way audio | NETDEV\_StopVoiceCom |
| Turn on microphone | NETDEV\_OpenMic |
| Turn off microphone | NETDEV\_CloseMic |
| Adjust microphone volume | NETDEV\_MicVolumeControl |
| Get microphone volume | NETDEV\_GetMicVolume |
| Get mute status | NETDEV\_GetMuteStatus |
| Set mute status | NETDEV\_SetMuteStatus |
| Turn on sound | NETDEV\_OpenSound |
| Adjust speaker volume | NETDEV\_SoundVolumeControl |
| Get speaker volume | NETDEV\_GetSoundVolume |
| Turn off sound | NETDEV\_CloseSound |
| Star audio data input service | NETDEV\_StartInputVoiceSrv |
| Stop audio data input service | NETDEV\_StopInputVoiceSrv |
| Input audio data | NETDEV\_InputVoiceData |
| Register the callback function to receive parsed audio data | NETDEV\_SetParseVoiceDataCB |
| Search recording files by file type and time | NETDEV\_FindFile |
| Get file info one by one | NETDEV\_FindNextFile |
| Stop searching file and release resource | NETDEV\_FindClose |
| Play recording file by name | NETDEV\_PlayBackByName |
| Play recording file by time | NETDEV\_PlayBackByTime |
| Control playback status | NETDEV\_PlayBackControl |
| Stop playback | NETDEV\_StopPlayBack |
| Download recording file by name | NETDEV\_GetFileByName |
| Download recording file by time | NETDEV\_GetFileByTime |
| Stop downloading recording file | NETDEV\_StopGetFile |
| Open local file | NETDEV\_OpenMediaFile |
| Play local file | NETDEV\_StartPlayMediaFile |
| Stop playing local file | NETDEV\_StopPlayMediaFile |
| Get total duration of media file | NETDEV\_GetMediaFileTime |
| Perform PTZ control (live view not started) | NETDEV\_PTZControl\_Other |
| Get preset position list | NETDEV\_GetPTZPresetList |
| Operate presets (no need to start preview) | NETDEV\_PTZPreset\_Other |
| Get preset patrol route | NETDEV\_PTZGetCruise |
| Operate preset patrol (no need to start preview) | NETDEV\_PTZCruise\_Other |
| Get recorded patrol route | NETDEV\_PTZGetTrackCruise |
| Operate recorded patrol (no need to start preview) | NETDEV\_PTZTrackCruise |
| Drag to zoom in/out (no need to start preview) | NETDEV\_PTZSelZoomIn\_Other |
| 3D positioning (no need to start preview) | NETDEV\_PTZ3DPosition\_Other |
| Get device configuration info | NETDEV\_GetDevConfig |
| Set device configuration info | NETDEV\_SetDevConfig |
| Change device name | NETDEV\_ModifyDeviceName |
| Get mapped port | NETDEV\_GetUpnpNatState |
| Set mapped port | NETDEV\_SetUpnpNatState |
| Get system time of device | NETDEV\_GetSystemTimeCfg |
| Set system time of device | NETDEV\_SetSystemTimeCfg |
| Restart device | NETDEV\_Reboot |
| Restore factory defaults. | NETDEV\_RestoreConfig |
| Start manual recording | NETDEV\_StartManualRecord |
| Stop manual recording | NETDEV\_StopManualRecord |
| Search device alarm by time | NETDEV\_FindAlarmInfoList |
| Get alarm info one by one | NETDEV\_FindNextAlarmInfo |
| Stop searching alarm and release resource | NETDEV\_FindCloseAlarmInfo |
| Get all info of all users | NETDEV\_GetUserDetailList |
| Add user info | NETDEV\_CreateUser |
| Modify user info | NETDEV\_ModifyUser |
| Delete user info | NETDEV\_DeleteUser |
| Register the callback function to receive people counting statistics | NETDEV\_SetPassengerFlowStatisticCallBack |
| Register callback function to receive captured face image | NETDEV\_SetFaceSnapshotCallBack |
| Add person library info | NETDEV\_CreatePersonLibInfo |
| Modify person library info | NETDEV\_ModifyPersonLibInfo |
| Search all person library info | NETDEV\_FindPersonLibList |
| Get person library info one by one | NETDEV\_FindNextPersonLibInfo |
| Stop searching person library and release resource | NETDEV\_FindClosePersonLibList |
| Delete specified person library | NETDEV\_DeletePersonLibInfo |
| Add specified person info | NETDEV\_AddPersonInfo |
| Modify specified person info | NETDEV\_ModifyPersonInfo |
| Delete specified person info | NETDEV\_DeletePersonInfo |
| Search all face monitoring tasks | NETDEV\_FindPersonMonitorList |
| Get monitoring task info one by one | NETDEV\_FindNextPersonMonitorInfo |
| Stop searching monitoring task and release resource | NETDEV\_FindClosePersonMonitorList |
| Add a single face monitoring task | NETDEV\_AddPersonMonitorInfo |
| Delete a single face monitoring task | NETDEV\_DeletePersonMonitorInfo |
| Get info about a single face monitoring task | NETDEV\_GetPersonMonitorRuleInfo |
| Set a single face monitoring task | NETDEV\_SetPersonMonitorRuleInfo |
| Delete face monitoring tasks in batches | NETDEV\_BatchDeletePersonMonitorInfo |
| Get capacity info of all person libraries | NETDEV\_GetPersonLibCapacity |
| Search person info with criteria | NETDEV\_FindPersonInfoList |
| Get person info one by one | NETDEV\_FindNextPersonInfo |
| Stop searching person info and release resource | NETDEV\_FindClosePersonInfoList |
| Register the callback function to receive face recognition alarm | NETDEV\_SetPersonAlarmCallBack |
| Register callback function to receive structured alarm | NETDEV\_SetStructAlarmCallBack |
| LAPI alarm subscription | NETDEV\_SubscibeLapiAlarm |
| Cancel LAPI alarm subscription | NETDEV\_UnSubLapiAlarm |

## NVR

|  |  |
| --- | --- |
| **Function** | **Interface** |
| Get video channel list | NETDEV\_QueryVideoChlDetailList |
| Get video channel list | NETDEV\_QueryVideoChlDetailListEx |
| Start live view | NETDEV\_RealPlay |
| Stop live view | NETDEV\_StopRealPlay |
| Register the callback function to receive raw streams | NETDEV\_SetPlayDataCallBack |
| Register the callback function to receive assembled live stream data | NETDEV\_SetPlayParseCB |
| Register the callback function to receive the decoded video stream data | NETDEV\_SetPlayDecodeVideoCB |
| Register the callback function to receive the assembled audio stream data | NETDEV\_SetPlayParseAudioCB |
| Register the callback function to receive the decoded audio stream data | NETDEV\_SetPlayDecodeAudioCB |
| Register the stream callback function to set the callback of displayed data | NETDEV\_SetPlayDisplayCB |
| Get bit rate of window | NETDEV\_GetBitRate |
| Get frame rate of window | NETDEV\_GetFrameRate |
| Get encoding format of window | NETDEV\_GetVideoEncodeFmt |
| Get video resolution | NETDEV\_GetResolution |
| Get packet loss rate of window | NETDEV\_GetLostPacketRate |
| Reset packet loss rate of window | NETDEV\_ResetLostPacketRate |
| Set image display scale | NETDEV\_SetRenderScale |
| Get image parameters (current image parameters only) | NETDEV\_GetVideoEffect |
| Adjust current image | NETDEV\_SetVideoEffect |
| Set digital zoom | NETDEV\_SetDigitalZoom |
| Set fluency priority | NETDEV\_SetPictureFluency |
| Start two-way audio | NETDEV\_StartVoiceCom |
| Stop two-way audio | NETDEV\_StopVoiceCom |
| Turn on microphone | NETDEV\_OpenMic |
| Turn off microphone | NETDEV\_CloseMic |
| Adjust microphone volume | NETDEV\_MicVolumeControl |
| Get microphone volume | NETDEV\_GetMicVolume |
| Get mute status | NETDEV\_GetMuteStatus |
| Set mute status | NETDEV\_SetMuteStatus |
| Turn on sound | NETDEV\_OpenSound |
| Adjust speaker volume | NETDEV\_SoundVolumeControl |
| Get speaker volume | NETDEV\_GetSoundVolume |
| Turn off sound | NETDEV\_CloseSound |
| Star audio data input service | NETDEV\_StartInputVoiceSrv |
| Stop audio data input service | NETDEV\_StopInputVoiceSrv |
| Input audio data | NETDEV\_InputVoiceData |
| Register the callback function to receive parsed audio data | NETDEV\_SetParseVoiceDataCB |
| Live view snapshot | NETDEV\_CapturePicture |
| Snapshot (not in preview) | NETDEV\_CaptureNoPreview |
| Generate a keyframe dynamically | NETDEV\_MakeKeyFrame |
| Local recording | NETDEV\_SaveRealData |
| Stop local recording | NETDEV\_StopSaveRealData |
| Search recording files by file type and time | NETDEV\_FindFile |
| Get file info one by one | NETDEV\_FindNextFile |
| Stop searching file and release resource | NETDEV\_FindClose |
| Play recording file by name | NETDEV\_PlayBackByName |
| Play recording file by time | NETDEV\_PlayBackByTime |
| Control playback status | NETDEV\_PlayBackControl |
| Stop playback | NETDEV\_StopPlayBack |
| Download recording file by name | NETDEV\_GetFileByName |
| Download recording files by time | NETDEV\_GetFileByTime |
| Stop downloading recording file | NETDEV\_StopGetFile |
| Open local file | NETDEV\_OpenMediaFile |
| Play local file | NETDEV\_StartPlayMediaFile |
| Stop playing local file | NETDEV\_StopPlayMediaFile |
| Get total duration of media file | NETDEV\_GetMediaFileTime |
| Perform PTZ control (live view not started) | NETDEV\_PTZControl\_Other |
| Get preset position list | NETDEV\_GetPTZPresetList |
| Operate presets (no need to start preview) | NETDEV\_PTZPreset\_Other |
| Get preset patrol route | NETDEV\_PTZGetCruise |
| Operate preset patrol (no need to start preview) | NETDEV\_PTZCruise\_Other |
| Get recorded patrol route | NETDEV\_PTZGetTrackCruise |
| Operate recorded patrol (no need to start preview) | NETDEV\_PTZTrackCruise |
| Drag to zoom in/out (no need to start preview) | NETDEV\_PTZSelZoomIn\_Other |
| 3D positioning (no need to start preview) | NETDEV\_PTZ3DPosition\_Other |
| Get device configuration info | NETDEV\_GetDevConfig |
| Set device configuration info | NETDEV\_SetDevConfig |
| Change device name | NETDEV\_ModifyDeviceName |
| Get mapped port | NETDEV\_GetUpnpNatState |
| Set mapped port | NETDEV\_SetUpnpNatState |
| Get system time of device | NETDEV\_GetSystemTimeCfg |
| Set system time of device | NETDEV\_SetSystemTimeCfg |
| Restart device | NETDEV\_Reboot |
| Restore factory defaults | NETDEV\_RestoreConfig |
| Start manual recording | NETDEV\_StartManualRecord |
| Stop manual recording | NETDEV\_StopManualRecord |
| Get all info of all users | NETDEV\_GetUserDetailList |
| Add user info | NETDEV\_CreateUser |
| Modify user info | NETDEV\_ModifyUser |
| Delete user info | NETDEV\_DeleteUser |
| Search log info by log type and time | NETDEV\_FindLogInfoList |
| Get log info one by one | NETDEV\_FindNextLogInfo |
| Stop searching log and release resource | NETDEV\_FindCloseLogInfo |
| Get people counting list | NETDEV\_FindTrafficStatisticInfoList |
| Get people counting info one by one | NETDEV\_FindNextTrafficStatisticInfo |
| Stop searching and release resource | NETDEV\_FindCloseTrafficStatisticInfo |
| Get people counting progress | NETDEV\_GetTrafficStatisticProgress |
| Start multi-channel people counting | NETDEV\_StartMultiTrafficStatistic |
| Stop multi-channel people counting | NETDEV\_StopTrafficStatistic |
| Add person library info | NETDEV\_CreatePersonLibInfo |
| Modify person library info | NETDEV\_ModifyPersonLibInfo |
| Search all person library info | NETDEV\_FindPersonLibList |
| Get person library info one by one | NETDEV\_FindNextPersonLibInfo |
| Stop searching person library and release resource | NETDEV\_FindClosePersonLibList |
| Delete specified person library | NETDEV\_DeletePersonLibInfo |
| Add specified person info | NETDEV\_AddPersonInfo |
| Modify specified person info | NETDEV\_ModifyPersonInfo |
| Delete specified person info | NETDEV\_DeletePersonInfo |
| Search all face monitoring tasks | NETDEV\_FindPersonMonitorList |
| Get monitoring task info one by one | NETDEV\_FindNextPersonMonitorInfo |
| Stop searching monitoring task and release resource | NETDEV\_FindClosePersonMonitorList |
| Add a single face monitoring task | NETDEV\_AddPersonMonitorInfo |
| Delete a single face monitoring task | NETDEV\_DeletePersonMonitorInfo |
| Delete face monitoring tasks in batches | NETDEV\_BatchDeletePersonMonitorInfo |
| Get info about a single face monitoring task | NETDEV\_GetPersonMonitorRuleInfo |
| Set a single face monitoring task | NETDEV\_SetPersonMonitorRuleInfo |
| Get capacity info of all person libraries | NETDEV\_GetPersonLibCapacity |
| Search person info with criteria | NETDEV\_FindPersonInfoList |
| Get person info one by one | NETDEV\_FindNextPersonInfo |
| Stop searching person info and release resource | NETDEV\_FindClosePersonInfoList |
| Register the callback function to receive face recognition alarm | NETDEV\_SetPersonAlarmCallBack |
| Register the callback function to receive structured alarm | NETDEV\_SetStructAlarmCallBack |
| LAPI alarm subscription | NETDEV\_SubscibeLapiAlarm |
| Cancel LAPI alarm subscription | NETDEV\_UnSubLapiAlarm |

## VMS

| **Function** | **Interface** |
| --- | --- |
| Get specified organization info | NETDEV\_GetOrgInfo |
| Batch add devices | NETDEV\_BatchAddDevice |
| Batch delete devices | NETDEV\_BatchDelDevice |
| Get device list by device type | NETDEV\_FindDevList |
| Get device info one by one | NETDEV\_FindNextDevInfo |
| Stop searching device info and release resource | NETDEV\_FindCloseDevInfo |
| Get device list by device ID or channel type | NETDEV\_FindDevChnList |
| Get device channel info one by one | NETDEV\_FindNextDevChn |
| Stop searching device channel info and release resource | NETDEV\_FindCloseDevChn |
| Start live view | NETDEV\_RealPlay |
| Stop live view | NETDEV\_StopRealPlay |
| Register the callback function to receive raw streams | NETDEV\_SetPlayDataCallBack |
| Register the callback function to receive assembled live stream data | NETDEV\_SetPlayParseCB |
| Register the callback function to receive the decoded video stream data | NETDEV\_SetPlayDecodeVideoCB |
| Register the callback function to receive the assembled audio stream data | NETDEV\_SetPlayParseAudioCB |
| Register the callback function to receive the decoded audio stream data | NETDEV\_SetPlayDecodeAudioCB |
| Register the stream callback function to set the callback of displayed data | NETDEV\_SetPlayDisplayCB |
| Get bit rate of window | NETDEV\_GetBitRate |
| Get frame rate of window | NETDEV\_GetFrameRate |
| Get encoding format of window | NETDEV\_GetVideoEncodeFmt |
| Get video resolution | NETDEV\_GetResolution |
| Get packet loss rate of window | NETDEV\_GetLostPacketRate |
| Reset packet loss rate of window | NETDEV\_ResetLostPacketRate |
| Set image display scale | NETDEV\_SetRenderScale |
| Get image parameters (current image parameters only) | NETDEV\_GetVideoEffect |
| Adjust current image | NETDEV\_SetVideoEffect |
| Set digital zoom | NETDEV\_SetDigitalZoom |
| Set fluency priority | NETDEV\_SetPictureFluency |
| Start two-way audio | NETDEV\_StartVoiceCom |
| Stop two-way audio | NETDEV\_StopVoiceCom |
| Turn on microphone | NETDEV\_OpenMic |
| Turn off microphone | NETDEV\_CloseMic |
| Adjust microphone volume | NETDEV\_MicVolumeControl |
| Get microphone volume | NETDEV\_GetMicVolume |
| Get mute status | NETDEV\_GetMuteStatus |
| Set mute status | NETDEV\_SetMuteStatus |
| Turn on sound | NETDEV\_OpenSound |
| Adjust speaker volume | NETDEV\_SoundVolumeControl |
| Get speaker volume | NETDEV\_GetSoundVolume |
| Turn off sound | NETDEV\_CloseSound |
| Star audio data input service | NETDEV\_StartInputVoiceSrv |
| Stop audio data input service | NETDEV\_StopInputVoiceSrv |
| Input audio data | NETDEV\_InputVoiceData |
| Register the callback function to receive parsed audio data | NETDEV\_SetParseVoiceDataCB |
| Get channel status in audio broadcast group | NETDEV\_GetVoiceBroadcastChlStatus |
| Add audio broadcast group | NETDEV\_CreateVoiceBroadcastGroup |
| Modify audio broadcast group | NETDEV\_ModifyVoiceBroadcastGroup |
| Start audio broadcast | NETDEV\_StartVoiceBroadcast |
| Stop audio broadcast | NETDEV\_StopVoiceBroadcast |
| Live view snapshot | NETDEV\_CapturePicture |
| Generate a keyframe dynamically | NETDEV\_MakeKeyFrame |
| Local recording | NETDEV\_SaveRealData |
| Stop local recording | NETDEV\_StopSaveRealData |
| Search recording files by file type and time | NETDEV\_FindFile |
| Get file info one by one | NETDEV\_FindNextFile |
| Stop searching file and release resource | NETDEV\_FindClose |
| Play recording file by time | NETDEV\_PlayBackByTime |
| Control playback status | NETDEV\_PlayBackControl |
| Stop playback | NETDEV\_StopPlayBack |
| Download recording file by time | NETDEV\_GetFileByTime |
| Stop downloading recording file | NETDEV\_StopGetFile |
| Open local file | NETDEV\_OpenMediaFile |
| Play local file | NETDEV\_StartPlayMediaFile |
| Stop playing local file | NETDEV\_StopPlayMediaFile |
| Get total duration of media file | NETDEV\_GetMediaFileTime |
| User logs in to a cloud account | NETDEV\_LoginCloud |
| Perform PTZ control (live view not started) | NETDEV\_PTZControl\_Other |
| Get preset position list | NETDEV\_GetPTZPresetList |
| Operate presets (no need to start preview) | NETDEV\_PTZPreset\_Other |
| Get preset patrol route | NETDEV\_PTZGetCruise |
| Operate preset patrol (no need to start preview) | NETDEV\_PTZCruise\_Other |
| Get recorded patrol route | NETDEV\_PTZGetTrackCruise\_V30 |
| Operate recorded patrol (no need to start preview) | NETDEV\_PTZTrackCruise\_V30 |
| Drag to zoom in/out (no need to start preview) | NETDEV\_PTZSelZoomIn\_Other |
| 3D positioning (no need to start preview) | NETDEV\_PTZ3DPosition\_Other |
| Get people counting list | NETDEV\_FindTrafficStatisticInfoList |
| Get people counting info one by one | NETDEV\_FindNextTrafficStatisticInfo |
| Stop searching and release resource | NETDEV\_FindCloseTrafficStatisticInfo |
| Get people counting progress | NETDEV\_GetTrafficStatisticProgress |
| Start multi-channel people counting | NETDEV\_StartMultiTrafficStatistic |
| Stop multi-channel people counting | NETDEV\_StopTrafficStatistic |
| Get access control permission group list | NETDEV\_FindACSPermissionGroupList |
| Get the next record | NETDEV\_FindNextACSPermissionGroupInfo |
| Stop searching records and release resource | NETDEV\_FindCloseACSPermissionGroupList |
| Add access control permission group | NETDEV\_AddACSPersonPermissionGroup |
| Modify access control permission group | NETDEV\_ModifyACSPersonPermissionGroup |
| Delete access control permission group | NETDEV\_DeleteACSPersonPermissionGroup |
| Get a single access control permission group | NETDEV\_GetSinglePermGroupInfo |
| Sign out visitor | NETDEV\_ACSPersonSignout |
| Get permission info of specified person | NETDEV\_GetACSPersonPermission |
| Set permission of specified person | NETDEV\_SetACSPersonPermission |
| Get entry/exit records | NETDEV\_FindACSAttendanceLogList |
| Get the next entry/exit record | NETDEV\_FindNextACSAttendanceLog |
| Stop searching entry/exit record and release resource | NETDEV\_FindCloseACSAttendanceLogList |
| Add visitor blocklist | NETDEV\_AddACSPersonBlackList |
| Delete visitor blocklist | NETDEV\_DeleteACSPersonBlackList |
| Modify visitor blocklist | NETDEV\_ModifyACSPersonBlackList |
| Get specified visitor blocklist | NETDEV\_GetACSPersonBlackList |
| Get visitor blocklist list | NETDEV\_FindACSPersonBlackList |
| Get the next record | NETDEV\_FindNextACSPersonBlackListInfo |
| Stop searching records and release resource | NETDEV\_FindCloseACSPersonBlackList |
| Access channel control | NETDEV\_DoorCtrl |
| Control doors in batches | NETDEV\_DoorBatchCtrl |
| Access person control | NETDEV\_ACSPersonCtrl |
| Add person info in batches | NETDEV\_AddACSPersonList |
| Delete access control person info in batches | NETDEV\_DeleteACSPersonList |
| Get access control person list | NETDEV\_FindACSPersonList |
| Get the next access control person info | NETDEV\_FindNextACSPersonInfo |
| Stop searching access control person list and release resource | NETDEV\_FindCloseACSPersonInfo |
| Get info about access control card bound with specified person | NETDEV\_GetACSPersonCardInfo |
| Bind access control card with specified person (i.e. open card) | NETDEV\_AddACSPersonCardInfo |
| Delete access control card bound with specified person (i.e. return card) | NETDEV\_DeleteACSPersonCardInfo |
| Get visitor record | NETDEV\_FindACSVisitLogList |
| Get the next visitor record | NETDEV\_FindNextACSVisitLog |
| Stop searching visitor record and release resource | NETDEV\_FindCloseACSVisitLog |
| Callback function for receiving alarm messages | NETDEV\_SetAlarmFGCallBack |
| Configure third-party payment order | NETDEV\_IssuePayOrder |
| Register callback function to receive parking lot event | NETDEV\_SetParkEventCallBack |
| Get parking lot info in batches | NETDEV\_GetParkInfoList |
| Get all entrance and exit info of a parking lot | NETDEV\_GetParkEntranceInfoList |
| Get all entrance and exit lane info of a parking lot | NETDEV\_GetParkLaneInfoList |
| Add person library info | NETDEV\_CreatePersonLibInfo |
| Modify person library info | NETDEV\_ModifyPersonLibInfo |
| Search all person library info | NETDEV\_FindPersonLibList |
| Get person library info one by one | NETDEV\_FindNextPersonLibInfo |
| Stop searching face library and release resource | NETDEV\_FindClosePersonLibList |
| Delete specified person library | NETDEV\_DeletePersonLibInfo |
| Add specified person info | NETDEV\_AddPersonInfo |
| Modify specified person info | NETDEV\_ModifyPersonInfo |
| Batch delete person info | NETDEV\_DeletePersonInfoList |
| Search all face monitoring tasks | NETDEV\_FindPersonMonitorList |
| Get monitoring task info one by one | NETDEV\_FindNextPersonMonitorInfo |
| Stop searching monitoring task and release resource | NETDEV\_FindClosePersonMonitorList |
| Add a single face monitoring task | NETDEV\_AddPersonMonitorInfo |
| Delete a single face monitoring task | NETDEV\_DeletePersonMonitorInfo |
| Get info about a single face monitoring task | NETDEV\_GetPersonMonitorRuleInfo |
| Set a single face monitoring task | NETDEV\_SetPersonMonitorRuleInfo |
| Get capacity info of all person libraries | NETDEV\_GetPersonLibCapacity |
| Search person info with criteria | NETDEV\_FindPersonInfoList |
| Get person info one by one | NETDEV\_FindNextPersonInfo |
| Stop searching person info and release resource | NETDEV\_FindClosePersonInfoList |
| Register the callback function to receive face recognition alarm | NETDEV\_SetPersonAlarmCallBack |
| Register the callback function to receive structured alarm info | NETDEV\_SetStructAlarmCallBack |
| Delete face monitoring tasks in batches | NETDEV\_BatchDeletePersonMonitorInfo |
| Get face recognition record | NETDEV\_FindFaceRecordDetailList |
| Get face recognition record info one by one | NETDEV\_FindNextFaceRecordDetail |
| Stop searching face recognition record and release resource | NETDEV\_FindCloseFaceRecordDetail |
| Get face image info of a single face recognition record | NETDEV\_GetFaceRecordImageInfo |
| Get system image info | NETDEV\_GetSystemPicture |
| Register callback function to receive plate recognition alarm | NETDEV\_SetVehicleAlarmCallBack |
| Search detailed vehicle member info with criteria | NETDEV\_FindVehicleMemberDetailList |
| Get vehicle member info one by one | NETDEV\_FindNextVehicleMemberDetail |
| Stop searching vehicle member and release resource | NETDEV\_FindCloseVehicleMemberDetail |
| Add vehicle members in batches | NETDEV\_AddVehicleMemberList |
| Delete vehicle members in batches | NETDEV\_DelVehicleMemberList |
| Delete specified vehicle member | NETDEV\_DelVehicleInfo |
| Get detailed info about a single vehicle member | NETDEV\_GetVehicleMemberInfo |
| Modify vehicle info in specified vehicle library | NETDEV\_ModifyVehicleMemberInfo |
| Get all monitoring tasks of vehicle recognition | NETDEV\_FindVehicleMonitorList |
| Get vehicle monitoring task info one by one | NETDEV\_FindNextVehicleMonitorInfo |
| Stop searching vehicle monitoring task and release resource | NETDEV\_FindCloseVehicleMonitorList |
| Add a single vehicle monitoring task | NETDEV\_AddVehicleMonitorInfo |
| Delete vehicle monitoring tasks in batches | NETDEV\_DeleteVehicleMonitorInfo |
| Get info about a single vehicle monitoring task | NETDEV\_GetVehicleMonitorInfo |
| Set a single vehicle monitoring task | NETDEV\_SetVehicleMonitorInfo |
| Search detailed vehicle recognition record info with criteria | NETDEV\_FindVehicleRecordInfoList |
| Get vehicle recognition record info one by one | NETDEV\_FindNextVehicleRecordInfo |
| Stop searching vehicle recognition record and release resource | NETDEV\_FindCloseVehicleRecordList |
| Get vehicle image info of a single vehicle recognition record | NETDEV\_GetVehicleRecordImageInfo |
| Assign vehicle members in batches to specified vehicle library | NETDEV\_AddVehicleLibMember |
| Remove vehicle members in batches from specified vehicle library | NETDEV\_DeleteVehicleLibMember |
| Get all vehicle library lists | NETDEV\_FindVehicleLibList |
| Get vehicle library info one by one | NETDEV\_FindNextVehicleLibInfo |
| Stop searching vehicle library and release resource | NETDEV\_FindCloseVehicleLibList |
| Add a single vehicle library | NETDEV\_AddVehicleLibInfo |
| Modify specified vehicle library | NETDEV\_ModifyVehicleLibInfo |
| Delete specified vehicle library | NETDEV\_DeleteVehicleLibInfo |
| Add time template | NETDEV\_AddTimeTemplate |
| Delete specified time templates in batches | NETDEV\_BatchDeleteTimeTemplate |
| Modify specified time template | NETDEV\_ModifyTimeTemplate |
| Get info about specified time template | NETDEV\_GetTimeTemplate |
| Get time templates of specified template type in batches | NETDEV\_FindTimeTemplateByTypeList |
| Get time template info by type one by one | NETDEV\_FindNextTimeTemplateByTypeInfo |
| Stop searching time template by type and release resource | NETDEV\_FindCloseTimeTemplateByTypeList |
| Subscribe to VCA events | NETDEV\_SubscribeSmart |
| Cancel subscription to VCA events | NETDEV\_UnsubscribeSmart |
| LAPI alarm subscription | NETDEV\_SubscibeLapiAlarm |
| Cancel LAPI alarm subscription | NETDEV\_UnSubLapiAlarm |

# Help

## Programming Environment Setup

### Development and Compilation Environment on Windows

* **Directories in the SDK**

The SDK contains the following folders:

include: contains the required header files;

lib: contains the required library files;

dll: contains the required dynamic libraries;

doc: contains the SDK User Manual;

demo: contains the MFC and C# source code for the demo;

demo: contains the executable program of demo.

* **Set compiler options of the integrated development environment**

Take VS2008 as an example. SDK path: C:\NetDEVSDK;

(1) Choose **Project** > **Properties** to open the **Property** page. Choose **Property Configuration** > **C/C++** > **General**. Enter **C:\NetDEVSDK\include** in **Additional Include Directories**.

(2) Choose **Property Configuration** > **Linker** > **General**. Enter **C:\NetDEVSDK\lib** in **Additional Library Directories**.

(3) Choose **Property Configuration** > **Linker** > **Input**, enter the required library files in **Additional Dependencies**.

* **NOTE!**

Make sure you keep all the dynamic library files, configuration files, and application programs of the SDK under the same directory. For example, keep all the files under **dll** in the same directory as NetDEVSDK.exe.

The name of the exported SDK log is “netdevsdk.log”.

* **Dependent Libraries and Header Files**

|  |  |  |
| --- | --- | --- |
| Header file | Library file (Windows) | Remarks |
| NetDEVSDK.h | NetDEVSDK.lib |  |

### Development and Compilation Environment on Linux

* **Directories in the SDK**

include: contains the required header files;

lib: contains the required Linux library;

doc: contains the SDK User Manual;

demo: contains the demo source code

* **Configure environment variables**

(1) Use the **cd** command to access the lib directory, copy the path of the lib directory;

(2) Add the lib directory path to the vi /etc/ld.so.conf file, save and exit.

(3) Execute the ldconfig command to make the environment variable take effect;

### Development and Compilation Environment on Android

* **Directories in the SDK**

include: contains the required header files;

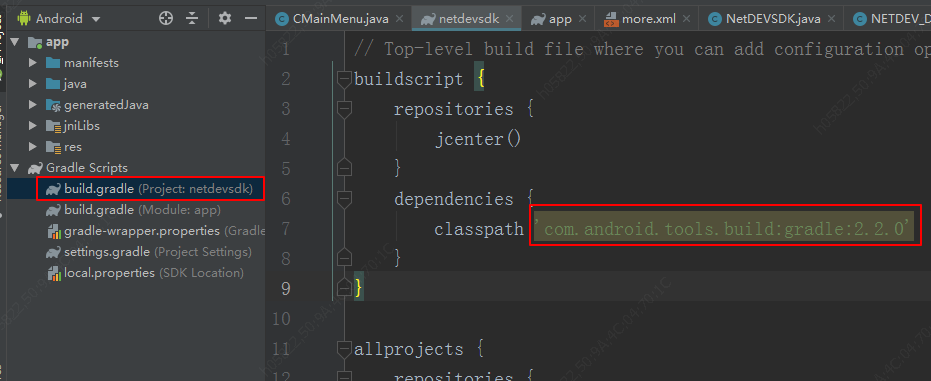
lib: contains the required Android libraries (32-bit and 64-bit);

doc: contains the SDK User Manual;

demo: contains the source code of Android Demo;

* **Demo configuration**

(1)Configure Build.gradle(Project:netdevsdk), and change the gradle version to the user’s version number.



(2)Configure Build.gradle(Module:app), and change the SDK version to the user’s version number.



(3)Configure local.properties, change the SDK path to the user’s local path.

### Development and Compilation Environment on iOS

* **Directories in the SDK**

include: contains the required header files;

lib: contains the required Linux library;

doc: contains the SDK User Manual;

demo: contains the demo source code

* **Xcode configuration**

1. Add SDK library and other dependent libraries. Copy the lib files extracted from the package to the project path, and add the SDK static library files in Link Binary With Libraries of Build Phases. Add dependent files for iPhone SDK. SDK static library files: all the .a files in the lib directory. Other files: libiconv.tbd, libstdc++.tbd, libz.tbd, libc++.tbd.
2. Configure build settings. Set **Enable Bitcode** to **NO**, set **Enable Testability** to **NO**;

## Precautions

* Interfaces and structs suffixed with \_V30 are preferred.

V30 has optimized functions and structs to provide better compatibility. It is recommended to use the latest version of functions and structs.

* To update the SDK version, you must update all the library files and header files.
* Unless otherwise stated, all interfaces must be called after the SDK is initialized.
* When capturing and downloading images, the directory where the path of the incoming file is located must exist; otherwise, the call will fail.
* Before using the NETPluginSDK, you must install it as administrator.

## FAQ

1. What is the SDK login port?

* Same as the web service port of the device. For example, if the web service port is 80, then the default SDK port is also 80.

1. How to tell if playback/download has ended?

* When the upper layer calls SDK interfaces for playback/download, the start and end times will be given.
* Call SDK interfaces to get the current playback/download timestamp.
* Compare the end time passed in with the current timestamp. The download has ended if the current timestamp is greater than or equal to the end time passed in.

1. Does the SDK support connection of third-party devices?

* No.

1. Streaming by SDK succeeded in LAN; after port mapping to WAN, streaming by Web succeeded and login by SDK also succeeded, but why streaming by SDK failed?

* Differences between streaming by Web and SDK?
* SDK uses the HTTP port for login and uses the RTSP port for streaming.
* Streaming by SDK failed because the RTSP port is blocked. Use commands to test the HTTP and RSTP ports separately to check port mapping. For example, the figure below shows the port is disconnected: telnet 192.168.3.129 50554

1. Blurry image or green screen is displayed after streaming by SDK succeeded in Windows.

* Probably it is because the graphics card driver is too old or the DirectX function is disabled. Use the DirectX diagnostic tool (press Win+R and run dxdiag). The normal display is as shown below.
* Common solution: Upgrade the graphics card driver.

1. Why face images are not received by registering the NETDEV\_SetFaceSnapshotCallBack callback for the camera's face capturing function?

* Enable the face capture function in IE.
* Disable firewall on your computer.
* Configure the PC address and port for the smart server (SDK listens to port 54000 by default), as shown below.

1. Failed to get snapshots using the snapshot interface (non-preview).

* This function needs the camera to support it. On the camera's web interface, check whether snapshot can be enabled, as shown below:

1. Failed to receive alarms using the SDK.

* Disable firewall on your computer.
* On the device's web interface, check if the logs contain alarm records to make sure the alarms have actually been triggered. For example, check alarm records on the NVR:

1. Link errors are reported during compile time when integrating with SDK libraries on Linux.

* In the **/etc/ld.so.conf** file, add the absolute path to the SDK library.
* Run the **ldconfig** command.

1. The application cannot start when I run MFC Demo. Is it because a side-by-side configuration error?

* You need to install VC2800 runtime library. MFC operation needs the system environment. Download the corresponding vcredit from the Internet and install it.

1. Cannot load C# demo source code into SDK libraries during compile time and runtime.

* Check whether all the SDK DLLs are included in the program's output path.
* Check whether project configuration is correct, whether it matches the bit number of the downloaded version. The figure below shows the settings of the 32-bit version.

1. Chinese characters are displayed incorrectly in the data obtained through the SDK.

* Strings obtained through the SDK are all in UTF8 format, so Chinese characters need to be converted to display properly.

1. Q1: Things that you need to know before using NETDEV\_SetDevConfig () to configure parameters.

* NETDEV\_SetDevConfig () requires the input to be a struct with complete assignment, otherwise, errors are likely to occur. To avoid this problem, before you call NETDEV\_SetDevConfig (), call NETDEV\_GetDevConfig () to assign an initial value to the struct that needs modification, then modify the corresponding parameter, and then call NETDEV\_SetDevConfig ().

1. What is audio data forwarding? What is it mainly used for?

* Audio data forwarding is the network part function of two-way audio, that is, the part of receiving and sending audio data. It can call back device audio and can also call NETDEV\_InputVoiceData () to send audio data to the device.
* The source of audio data is more flexible than two-way audio and can be collected by users or from audio files.

1. Can the callback function of live view and two-way audio be set to NULL?

* Yes, in that case, live view and two-way audio are still normal, but users cannot handle data by themselves.

1. Functions of audio broadcast?

* Refers to sending audio data to multiple devices simultaneously but not processing audio data from the devices. If a device has only one audio input interface, then only one of following can take place at the same time: audio broadcast, two-way audio, and audio forwarding.

1. Interfaces that must be called for live view and the correct sequence of calling?

* Correct sequence: NETDEV\_Init——> NETDEV\_Login\_V30——> NETDEV\_RealPlay——> NETDEV\_StopRealPlay——> NETDEV\_Logout ——> NETDEV\_Cleanup.