**网络设备SDK用户手册**

声明

由于产品版本升级或其他原因，本手册内容会不定期进行更新。

本手册仅作为使用指导，本手册中的所有陈述、信息和建议不构成任何明示或暗示的担保。如有任何疑问或争议，请以我司最终解释为准。

本公司对使用本手册或使用本公司产品导致的任何特殊、附带、偶然或间接的损害不承担责任，包括但不限于商业利润损失、数据或文档丢失产生的损失，因遭受网络攻击、黑客攻击、病毒感染等造成的产品工作异常、信息泄露。

本书约定

本书采用各种醒目标志来表示在使用过程中应该特别注意的地方，这些标志的意义如下：

| 标志 | 意义 |
| --- | --- |
| \\10.220.3.114\共享文件夹\运作资料部\02-作者交稿\奚珍珍\02-资料美工事宜\资料图标WMF\资料图标-警告02.wmf | 警告。该标志后的注释需给予格外关注，不当的操作可能会对人身造成伤害 |
| \\10.220.3.114\共享文件夹\运作资料部\01-资料开发平台\01-随机资料模板\01-宇视资料模板（word2007）\01-随机资料模板\Note图标\资料图标-注意02.wmf | 注意。提醒操作中应注意的事项，不当的操作可能会导致产品损坏、数据丢失或功能异常 |
| \\10.220.3.114\共享文件夹\运作资料部\02-作者交稿\奚珍珍\02-资料美工事宜\资料图标WMF\资料图标-说明02.wmf | 说明。对产品操作使用相关信息进行提示、补充 |

[1. 内容简介 58](#_Toc88647037)

[1.1 前言 58](#_Toc88647038)

[1.2 阅读对象 58](#_Toc88647039)

[1.3 概述 58](#_Toc88647040)

[1.4 名词解释 59](#_Toc88647041)

[1.5 系统要求 59](#_Toc88647042)

[1.6 适用的产品 60](#_Toc88647043)

[2. 版本更新 60](#_Toc88647044)

[2.1 版本更新说明V2.5 60](#_Toc88647045)

[2.2 版本更新说明V2.6 61](#_Toc88647046)

[3. 编程指导 61](#_Toc88647047)

[3.1 编程导引 61](#_Toc88647048)

[3.2 SDK调用主要流程 62](#_Toc88647049)

[3.2.1 流程图 62](#_Toc88647050)

[3.2.2 NVR或者IPC登录示例代码 64](#_Toc88647051)

[3.2.3 VMS登录示例代码 65](#_Toc88647052)

[3.3 参数配置模块流程 68](#_Toc88647053)

[3.3.1 流程图 68](#_Toc88647054)

[3.3.2 图像配置示例代码 68](#_Toc88647055)

[3.4 回放及下载模块流程 71](#_Toc88647056)

[3.4.1 流程图 71](#_Toc88647057)

[3.4.2 录像查询及通过时间回放示例代码 71](#_Toc88647058)

[3.4.3 录像查询及通过时间下载示例代码 74](#_Toc88647059)

[3.4.4 录像查询及通过文件名下载示例代码 77](#_Toc88647060)

[3.5 实况模块流程 81](#_Toc88647061)

[3.5.1 流程图 81](#_Toc88647062)

[3.5.2 实况示例代码 81](#_Toc88647063)

[3.6 语音模块流程 84](#_Toc88647064)

[3.6.1 语音对讲模块流程 84](#_Toc88647065)

[3.6.2 语音转发模块流程 87](#_Toc88647066)

[3.7 维护模块流程 90](#_Toc88647067)

[3.7.1 流程图 90](#_Toc88647068)

[3.7.2 示例代码 90](#_Toc88647069)

[3.8 异常模块流程 93](#_Toc88647070)

[3.8.1 流程图 93](#_Toc88647071)

[3.8.2 示例代码 93](#_Toc88647072)

[3.9 告警模块流程 96](#_Toc88647073)

[3.9.1 流程图 96](#_Toc88647074)

[3.9.2 示例代码 96](#_Toc88647075)

[3.10 云台模块流程 99](#_Toc88647076)

[3.10.1 云台控制流程 99](#_Toc88647077)

[3.10.2 预置位和巡航模块流程 103](#_Toc88647078)

[3.11 智能模块流程 104](#_Toc88647079)

[3.11.1 人员模块告警流程 104](#_Toc88647080)

[3.11.2 客流量模块功能流程 113](#_Toc88647081)

[3.11.3 车辆模块功能流程 114](#_Toc88647082)

[3.11.4 结构化数据（过人过车）流程 120](#_Toc88647083)

[4. 基本接口定义 126](#_Toc88647084)

[4.1 SDK初始化 126](#_Toc88647085)

[4.1.1 SDK初始化 126](#_Toc88647086)

[4.1.2 SDK清理 126](#_Toc88647087)

[4.2 SDK本地功能 127](#_Toc88647088)

[4.2.1 设置日志大小和数量 127](#_Toc88647089)

[4.2.2 设置日志文件路径 129](#_Toc88647090)

[4.2.3 设置写入日志的标记 130](#_Toc88647091)

[4.2.4 获取SDK版本信息 131](#_Toc88647092)

[4.2.5 获取接口错误码 131](#_Toc88647093)

[4.2.6 注册异常回调函数 132](#_Toc88647094)

[4.2.7 设置最大解码路数 134](#_Toc88647095)

[4.2.8 设置保活等待间隔及次数 135](#_Toc88647096)

[4.2.9 设置接收超时时间 136](#_Toc88647097)

[4.2.10 设置本地监听地址 137](#_Toc88647098)

[4.3 设备搜索 138](#_Toc88647099)

[4.3.1 设置设备搜索回调函数 138](#_Toc88647100)

[4.3.2 设备搜索 139](#_Toc88647101)

[4.4 免登陆配置 141](#_Toc88647102)

[4.4.1 修改设备IP地址 141](#_Toc88647103)

[4.5 用户登录 142](#_Toc88647104)

[4.5.1 用户设备登录 142](#_Toc88647105)

[4.5.2 用户注销 143](#_Toc88647106)

[4.6 实时预览 144](#_Toc88647107)

[4.6.1 实时预览 144](#_Toc88647108)

[4.6.2 停止实时预览 146](#_Toc88647109)

[4.6.3 启动本地录像 147](#_Toc88647110)

[4.6.4 停止本地录像 148](#_Toc88647111)

[4.6.5 获取实况起流URL 149](#_Toc88647112)

[4.6.6 通过URL启动实况预览 151](#_Toc88647113)

[4.6.7 预览抓图 153](#_Toc88647114)

[4.6.8 非预览抓图 154](#_Toc88647115)

[4.6.9 非预览获取图片数据 156](#_Toc88647116)

[4.6.10 设置录像文件大小 158](#_Toc88647117)

[4.7 远程回放 159](#_Toc88647118)

[4.7.1 录像查询 159](#_Toc88647119)

[4.7.2 获取录像天数 162](#_Toc88647120)

[4.7.3 查询指定月份的录像分布信息 163](#_Toc88647121)

[4.7.4 获取回放URL 166](#_Toc88647122)

[4.7.5 根据URL回放录像文件 167](#_Toc88647123)

[4.7.6 按文件名回放录像文件 168](#_Toc88647124)

[4.7.7 按时间回放录像文件 169](#_Toc88647125)

[4.7.8 单帧回放 170](#_Toc88647126)

[4.7.9 回放控制 171](#_Toc88647127)

[4.7.10 停止回放 172](#_Toc88647128)

[4.7.11 按文件名下载录像文件 173](#_Toc88647129)

[4.7.12 按时间下载录像文件 175](#_Toc88647130)

[4.7.13 停止下载录像文件 176](#_Toc88647131)

[4.7.14 查询录像起止时间 177](#_Toc88647132)

[4.8 码流数据回调 178](#_Toc88647133)

[4.8.1 注册原始码流回调 178](#_Toc88647134)

[4.8.2 注册解码后音频数据回调 180](#_Toc88647135)

[4.8.3 注册解码后视频数据回调 182](#_Toc88647136)

[4.8.4 注册拼帧后音频数据回调 184](#_Toc88647137)

[4.8.5 注册拼帧后视频数据回调 186](#_Toc88647138)

[4.8.6 注册画图回调 188](#_Toc88647139)

[4.9 视频参数配置 190](#_Toc88647140)

[4.9.1 获取窗口丢包率 190](#_Toc88647141)

[4.9.2 重置窗口丢包率 192](#_Toc88647142)

[4.9.3 获取窗口码率 192](#_Toc88647143)

[4.9.4 获取窗口帧率 193](#_Toc88647144)

[4.9.5 获取窗口分辨率 194](#_Toc88647145)

[4.9.6 获取影像参数 195](#_Toc88647146)

[4.9.7 影像调节 196](#_Toc88647147)

[4.9.8 获取窗口编码格式 197](#_Toc88647148)

[4.9.9 动态产生一个关键帧 198](#_Toc88647149)

[4.9.10 设置数字放大 199](#_Toc88647150)

[4.9.11 设置元数据处理开关 200](#_Toc88647151)

[4.9.12 设置元数据处理参数 201](#_Toc88647152)

[4.9.13 设置视频流畅性 202](#_Toc88647153)

[4.9.14 设置视频图像显示比例 203](#_Toc88647154)

[4.10 语音功能 204](#_Toc88647155)

[4.10.1 开启语音对讲 204](#_Toc88647156)

[4.10.2 停止语音对讲 206](#_Toc88647157)

[4.10.3 开启语音数据转发 207](#_Toc88647158)

[4.10.4 语音数据转发 208](#_Toc88647159)

[4.10.5 停止语音数据转发 209](#_Toc88647160)

[4.10.6 开启语音广播 210](#_Toc88647161)

[4.10.7 停止语音广播 211](#_Toc88647162)

[4.10.8 开启语音广播组 212](#_Toc88647163)

[4.10.9 修改语音广播组 213](#_Toc88647164)

[4.10.10 获取语音广播组通道状态 214](#_Toc88647165)

[4.10.11 修改语音广播组通道状态 215](#_Toc88647166)

[4.10.12 开启扬声器声音 216](#_Toc88647167)

[4.10.13 获取扬声器音量 217](#_Toc88647168)

[4.10.14 设置扬声器音量 218](#_Toc88647169)

[4.10.15 关闭扬声器声音 220](#_Toc88647170)

[4.10.16 设置静音状态 220](#_Toc88647171)

[4.10.17 获取静音状态 222](#_Toc88647172)

[4.10.18 开启麦克风 223](#_Toc88647173)

[4.10.19 获取麦克风音量 224](#_Toc88647174)

[4.10.20 设置麦克风音量 225](#_Toc88647175)

[4.10.21 关闭麦克风 226](#_Toc88647176)

[4.11 本地播放 227](#_Toc88647177)

[4.11.1 打开本地文件 227](#_Toc88647178)

[4.11.2 开始播放本地文件 228](#_Toc88647179)

[4.11.3 停止播放本地文件 229](#_Toc88647180)

[4.11.4 获取本地文件播放总时长 230](#_Toc88647181)

[4.12 鱼眼矫正 231](#_Toc88647182)

[4.12.1 鱼眼码流判断 231](#_Toc88647183)

[4.12.2 设置5ePTZ显示模式 232](#_Toc88647184)

[4.12.3 鼠标操作模式 232](#_Toc88647185)

[4.12.4 获取鱼眼矫正参数 234](#_Toc88647186)

[4.12.5 设置模式和安装位置 235](#_Toc88647187)

[4.12.6 获取模式和安装位置 236](#_Toc88647188)

[4.13 手动录像 237](#_Toc88647189)

[4.13.1 获取手动录像状态 237](#_Toc88647190)

[4.13.2 开启手动录像 238](#_Toc88647191)

[4.13.3 停止手动录像 239](#_Toc88647192)

[4.14 云台控制 240](#_Toc88647193)

[4.14.1 云台3D定位 240](#_Toc88647194)

[4.14.2 无预览云台3D定位 241](#_Toc88647195)

[4.14.3 云台拉框放大 242](#_Toc88647196)

[4.14.4 无预览云台拉框放大 243](#_Toc88647197)

[4.14.5 云台拉框缩小 244](#_Toc88647198)

[4.14.6 无预览云台拉框缩小 245](#_Toc88647199)

[4.14.7 拉框放大.缩小 246](#_Toc88647200)

[4.14.8 无预览拉框放大.缩小 247](#_Toc88647201)

[4.14.9 云台控制操作 249](#_Toc88647202)

[4.14.10 无预览云台控制操作 250](#_Toc88647203)

[4.14.11 获取、设置云台守望信息 251](#_Toc88647204)

[4.14.12 云台看守位操作 252](#_Toc88647205)

[4.14.13 无预览云台看守位操作 253](#_Toc88647206)

[4.14.14 云台绝对坐标转动 255](#_Toc88647207)

[4.14.15 获取云台状态 256](#_Toc88647208)

[4.14.16 获取云台轨迹巡航路径 257](#_Toc88647209)

[4.14.17 获取云台轨迹巡航路径（包含循环轨迹ID） 258](#_Toc88647210)

[4.14.18 获取指定通道的巡航状态 259](#_Toc88647211)

[4.14.19 云台轨迹巡航操作 260](#_Toc88647212)

[4.14.20 无预览云台轨迹巡航操作 261](#_Toc88647213)

[4.14.21 获取云台预置位列表 262](#_Toc88647214)

[4.14.22 云台预置位操作 263](#_Toc88647215)

[4.14.23 无预览云台预置位操作 264](#_Toc88647216)

[4.14.24 云台预置位巡航操作 266](#_Toc88647217)

[4.14.25 无预览云台预置位巡航操作 267](#_Toc88647218)

[4.14.26 获取云台预置位巡航路径 268](#_Toc88647219)

[4.14.27 云台标定 269](#_Toc88647220)

[4.14.28 获取云台镜头当前视角值 270](#_Toc88647221)

[4.14.29 获取云台的经纬度信息 271](#_Toc88647222)

[4.14.30 设置云台的经纬度信息 272](#_Toc88647223)

[4.14.31 获取云台当前变倍倍数 273](#_Toc88647224)

[4.14.32 设置云台当前变倍倍数 274](#_Toc88647225)

[4.15 系统配置 275](#_Toc88647226)

[4.15.1 开启/关闭Telnet 275](#_Toc88647227)

[4.15.2 导出配置文件 276](#_Toc88647228)

[4.15.3 导入配置文件 277](#_Toc88647229)

[4.15.4 获取设备的配置信息 278](#_Toc88647230)

[4.15.5 设置设备的配置信息 279](#_Toc88647231)

[4.15.6 查询硬盘模式 280](#_Toc88647232)

[4.15.7 设置硬盘模式 281](#_Toc88647233)

[4.15.8 获取设备系统时间配置 282](#_Toc88647234)

[4.15.9 设置设备系统时间配置 283](#_Toc88647235)

[4.15.10 无需登录修改设备IP地址 284](#_Toc88647236)

[4.15.11 修改设备名称 285](#_Toc88647237)

[4.15.12 获取设备能力集 286](#_Toc88647238)

[4.15.13 通过设备类型查询设备列表 287](#_Toc88647239)

[4.15.14 逐个获取查找到的设备信息 288](#_Toc88647240)

[4.15.15 关闭查找设备信息，释放资源 289](#_Toc88647241)

[4.15.16 获取通道设备基本信息列表 290](#_Toc88647242)

[4.15.17 逐个获取通道设备基本信息 291](#_Toc88647243)

[4.15.18 关闭获取通道设备基本信息 292](#_Toc88647244)

[4.15.19 通过设备ID或通道类型查询通道信息列表 292](#_Toc88647245)

[4.15.20 逐个获取查找到的设备通道信息 293](#_Toc88647246)

[4.15.21 关闭查找设备通道信息，释放资源 295](#_Toc88647247)

[4.15.22 根据通道类型和通道ID获取通道详细信息 295](#_Toc88647248)

[4.15.23 根据通道类型和通道ID设置报警输出通道的信息 297](#_Toc88647249)

[4.15.24 获取通道类型 298](#_Toc88647250)

[4.15.25 获取通道设备基本信息 299](#_Toc88647251)

[4.15.26 获取设备基础信息 300](#_Toc88647252)

[4.15.27 获取设备信息 301](#_Toc88647253)

[4.15.28 查询设备详细信息 301](#_Toc88647254)

[4.15.29 获取设备定位信息 302](#_Toc88647255)

[4.15.30 获取wifi sniffer MAC地址信息 303](#_Toc88647256)

[4.15.31 查询视频通道信息列表 304](#_Toc88647257)

[4.15.32 查询视频通道信息列表 305](#_Toc88647258)

[4.15.33 查询视频通道信息 306](#_Toc88647259)

[4.15.34 导入音频文件 307](#_Toc88647260)

[4.15.35 删除音频文件 308](#_Toc88647261)

[4.15.36 获取音频文件 310](#_Toc88647262)

[4.15.37 查询音频输出参数 313](#_Toc88647263)

[4.15.38 设置音频输出参数 314](#_Toc88647264)

[4.16 告警抓图 315](#_Toc88647265)

[4.16.1 获取告警联动抓图URL 315](#_Toc88647266)

[4.16.2 保存告警抓图 318](#_Toc88647267)

[4.17 告警消息 320](#_Toc88647268)

[4.17.1 注册回调函数,接收报警消息等 320](#_Toc88647269)

[4.17.2 主动获取设备告警信息 322](#_Toc88647270)

[4.18 客流量 323](#_Toc88647271)

[4.18.1 获取客流量统计信息列表 323](#_Toc88647272)

[4.18.2 逐个查询获取到的客流量信息 324](#_Toc88647273)

[4.18.3 关闭查找，释放资源 325](#_Toc88647274)

[4.18.4 获取客流量统计 325](#_Toc88647275)

[4.18.5 清零客流量人数统计 326](#_Toc88647276)

[4.18.6 获取客流量上报时间间隔 327](#_Toc88647277)

[4.18.7 开始客流量统计查询 328](#_Toc88647278)

[4.18.8 停止客流量查询 329](#_Toc88647279)

[4.18.9 多通道开始客流量统计查询 330](#_Toc88647280)

[4.18.10 获取客流量统计进度 331](#_Toc88647281)

[4.18.11 注册客流量统计上报回调函数 332](#_Toc88647282)

[4.19 布控任务 334](#_Toc88647283)

[4.19.1 新增指定的人员信息 334](#_Toc88647284)

[4.19.2 删除指定的人员信息 335](#_Toc88647285)

[4.19.3 新增单个人脸布控任务 336](#_Toc88647286)

[4.19.4 删除单个人脸布控任务 338](#_Toc88647287)

[4.19.5 批量删除人脸布控任务 339](#_Toc88647288)

[4.19.6 创建人员库信息 340](#_Toc88647289)

[4.19.7 修改人员库信息 341](#_Toc88647290)

[4.19.8 删除指定的人员库 342](#_Toc88647291)

[4.19.9 查询单个人脸布控任务配置信息 343](#_Toc88647292)

[4.19.10 设置单个人脸布控任务配置信息 344](#_Toc88647293)

[4.19.11 查询所有人员库的容量信息 345](#_Toc88647294)

[4.19.12 条件查询人员信息 346](#_Toc88647295)

[4.19.13 查询所有已创建的人员库信息 350](#_Toc88647296)

[4.19.14 查询所有人脸布控任务 353](#_Toc88647297)

[4.19.15 新增单个车辆库信息 357](#_Toc88647298)

[4.19.16 删除指定的车辆库信息 358](#_Toc88647299)

[4.19.17 修改指定的车辆库信息 359](#_Toc88647300)

[4.19.18 批量添加车辆成员信息 360](#_Toc88647301)

[4.19.19 批量删除车辆成员信息 361](#_Toc88647302)

[4.19.20 删除指定车辆成员信息 362](#_Toc88647303)

[4.19.21 新增单个车辆布控任务 363](#_Toc88647304)

[4.19.22 批量删除车辆布控任务 364](#_Toc88647305)

[4.19.23 获取单个车辆成员详细信息 365](#_Toc88647306)

[4.19.24 修改指定车辆库中车辆信息 366](#_Toc88647307)

[4.19.25 查询单个车辆布控任务配置信息 367](#_Toc88647308)

[4.19.26 设置单个车辆布控任务配置信息 368](#_Toc88647309)

[4.19.27 向指定的车辆库中批量划归车辆成员 369](#_Toc88647310)

[4.19.28 批量取消指定的车辆库中车辆成员划归 370](#_Toc88647311)

[4.19.29 查询单个车辆识别记录的车辆图片信息 372](#_Toc88647312)

[4.19.30 查询全部车辆库信息列表 373](#_Toc88647313)

[4.19.31 条件查询车辆识别记录的详细信息 376](#_Toc88647314)

[4.19.32 查询车辆识别的所有布控任务 380](#_Toc88647315)

[4.19.33 条件查询车辆成员详细信息 383](#_Toc88647316)

[4.19.34 修改指定的人员信息 387](#_Toc88647317)

[4.19.35 批量删除人员信息 388](#_Toc88647318)

[4.20 数据透传 389](#_Toc88647319)

[4.20.1 建立透明通道 389](#_Toc88647320)

[4.20.2 通过透明通道向设备串口发送数据 391](#_Toc88647321)

[4.20.3 断开透明通道 393](#_Toc88647322)

[4.21 用户配置 394](#_Toc88647323)

[4.21.1 创建用户信息 394](#_Toc88647324)

[4.21.2 修改用户信息 395](#_Toc88647325)

[4.21.3 删除用户信息 396](#_Toc88647326)

[4.21.4 修改当前用户密码 397](#_Toc88647327)

[4.21.5 修改指定用户密码 398](#_Toc88647328)

[4.21.6 获取设备当前密码 399](#_Toc88647329)

[4.21.7 获取指定用户全部信息 400](#_Toc88647330)

[4.21.8 获取所有用户全部信息 401](#_Toc88647331)

[4.22 组织管理 402](#_Toc88647332)

[4.22.1 添加组织 402](#_Toc88647333)

[4.22.2 修改组织 403](#_Toc88647334)

[4.22.3 批量删除组织 404](#_Toc88647335)

[4.22.4 根据组织ID修改通道 405](#_Toc88647336)

[4.22.5 根据组织ID获取通道 406](#_Toc88647337)

[4.22.6 通过组织ID查询通道信息列表 407](#_Toc88647338)

[4.22.7 查询组织信息列表 410](#_Toc88647339)

[4.23 网络配置 414](#_Toc88647340)

[4.23.1 获取NAT类型 414](#_Toc88647341)

[4.23.2 获取映射端口 415](#_Toc88647342)

[4.23.3 设置映射端口 416](#_Toc88647343)

[4.24 解码设备 417](#_Toc88647344)

[4.24.1 获取能力集 417](#_Toc88647345)

[4.24.2 获取本地编码通道数量 418](#_Toc88647346)

[4.24.3 获取视频通道信息列表 419](#_Toc88647347)

[4.24.4 创建电视墙配置 420](#_Toc88647348)

[4.24.5 修改电视墙配置 421](#_Toc88647349)

[4.24.6 删除电视墙配置 422](#_Toc88647350)

[4.24.7 获取电视墙配置 423](#_Toc88647351)

[4.24.8 获取单个电视墙布局配置信息 427](#_Toc88647352)

[4.24.9 创建单个虚拟LED 428](#_Toc88647353)

[4.24.10 删除单个虚拟LED 429](#_Toc88647354)

[4.24.11 配置单个虚拟LED 430](#_Toc88647355)

[4.24.12 获取单个虚拟LED 431](#_Toc88647356)

[4.24.13 获取虚拟LED列表信息 432](#_Toc88647357)

[4.24.14 创建单个窗口配置信息 436](#_Toc88647358)

[4.24.15 批量开窗 437](#_Toc88647359)

[4.24.16 修改单个窗口配置信息 438](#_Toc88647360)

[4.24.17 批量修改窗口 439](#_Toc88647361)

[4.24.18 删除单个窗口配置信息 440](#_Toc88647362)

[4.24.19 电视墙窗口放大/还原 441](#_Toc88647363)

[4.24.20 批量关窗 442](#_Toc88647364)

[4.24.21 获取窗口配置列表 443](#_Toc88647365)

[4.24.22 获取单个窗口配置信息 447](#_Toc88647366)

[4.24.23 获取单个实况业务绑定信息 448](#_Toc88647367)

[4.24.24 绑定单个实况业务 449](#_Toc88647368)

[4.24.25 删除单个实况业务绑定 450](#_Toc88647369)

[4.24.26 获取场景下的实况业务绑定信息 451](#_Toc88647370)

[4.24.27 开始被动解码 455](#_Toc88647371)

[4.24.28 向被动解码通道发送数据 456](#_Toc88647372)

[4.24.29 停止被动解码 457](#_Toc88647373)

[4.24.30 业务流信息 458](#_Toc88647374)

[4.24.31 获取设备全部轮巡资源 462](#_Toc88647375)

[4.24.32 添加单个轮巡资源信息到设备 463](#_Toc88647376)

[4.24.33 修改单个轮巡资源详细信息 464](#_Toc88647377)

[4.24.34 删除单个轮巡资源信息 465](#_Toc88647378)

[4.24.35 获取单个轮巡资源详细信息 466](#_Toc88647379)

[4.24.36 创建单个场景信息 467](#_Toc88647380)

[4.24.37 修改场景信息 468](#_Toc88647381)

[4.24.38 删除场景信息 469](#_Toc88647382)

[4.24.39 获取当前场景的关联场景 470](#_Toc88647383)

[4.24.40 切换场景 471](#_Toc88647384)

[4.24.41 获取当前场景配置内容 473](#_Toc88647385)

[4.24.42 获取场景信息 474](#_Toc88647386)

[4.24.43 获取场景配置列表 475](#_Toc88647387)

[4.24.44 绑定轮巡资源到当前场景布局 478](#_Toc88647388)

[4.24.45 修改当前场景轮巡布局 479](#_Toc88647389)

[4.24.46 删除当前场景轮巡布局 481](#_Toc88647390)

[4.24.47 轮巡播放控制 482](#_Toc88647391)

[4.24.48 添加单个场景轮巡计划 483](#_Toc88647392)

[4.24.49 修改单个场景轮巡计划 484](#_Toc88647393)

[4.24.50 删除单个场景轮巡计划 485](#_Toc88647394)

[4.24.51 获取单个场景轮巡计划 486](#_Toc88647395)

[4.24.52 获取场景轮巡计划列表 487](#_Toc88647396)

[4.24.53 场景轮巡计划播放控制 491](#_Toc88647397)

[4.25 智能功能 492](#_Toc88647398)

[4.25.1 获取系统图片信息 492](#_Toc88647399)

[4.25.2 手动联动操作 493](#_Toc88647400)

[4.25.3 注册车牌识别上报回调函数 494](#_Toc88647401)

[4.25.4 注册人脸识别报警消息回调函数 495](#_Toc88647402)

[4.25.5 注册人脸抓拍上报回调函数 497](#_Toc88647403)

[4.25.6 注册热度图上报回调函数 499](#_Toc88647404)

[4.25.7 注册人脸报警消息回调函数 500](#_Toc88647405)

[4.25.8 注册结构化报警消息回调函数 502](#_Toc88647406)

[4.25.9 注册车牌识别报警回调函数 504](#_Toc88647407)

[4.25.10 订阅智能事件 505](#_Toc88647408)

[4.25.11 取消订阅智能事件 507](#_Toc88647409)

[4.25.12 LAPI告警订阅 508](#_Toc88647410)

[4.25.13 取消LAPI告警订阅 509](#_Toc88647411)

[4.25.14 人脸识别记录 510](#_Toc88647412)

[4.25.15 停车场 515](#_Toc88647413)

[4.25.16 时间模板 520](#_Toc88647414)

[4.25.17 门禁 528](#_Toc88647415)

[4.25.18 深度周界 566](#_Toc88647416)

[4.25.19 获取指定告警的关联数据 572](#_Toc88647417)

[4.25.20 注册图片报警回调函数 575](#_Toc88647418)

[4.25.21 人数统计 577](#_Toc88647419)

[4.26 系统维护 586](#_Toc88647420)

[4.26.1 查找日志信息 586](#_Toc88647421)

[4.26.2 查询设备告警信息 590](#_Toc88647422)

[4.26.3 查询告警日志 594](#_Toc88647423)

[4.26.4 设备重启 597](#_Toc88647424)

[4.26.5 设备恢复出厂设置 598](#_Toc88647425)

[4.27 结构体定义 599](#_Toc88647426)

[4.27.1 设备登录信息结构体 599](#_Toc88647427)

[4.27.2 安全登录信息结构体 600](#_Toc88647428)

[4.27.3 超时时间结构体 601](#_Toc88647429)

[4.27.4 设备发现的设备信息结构体 602](#_Toc88647430)

[4.27.5 播放器异常输出信息结构体 603](#_Toc88647431)

[4.27.6 设备地址信息结构体 604](#_Toc88647432)

[4.27.7 实况预览参数结构体 606](#_Toc88647433)

[4.27.8 解码后音频数据结构体 607](#_Toc88647434)

[4.27.9 音频参数结构体 608](#_Toc88647435)

[4.27.10 通道操作信息结构体 609](#_Toc88647436)

[4.27.11 通道列表结构体 609](#_Toc88647437)

[4.27.12 语音广播组批处理信息结构体 610](#_Toc88647438)

[4.27.13 语音广播通道信息结构体 611](#_Toc88647439)

[4.27.14 语音广播组通道信息结构体 611](#_Toc88647440)

[4.27.15 语音广播组控制信息结构体 612](#_Toc88647441)

[4.27.16 音频数据结构体 613](#_Toc88647442)

[4.27.17 图像数据结构体 614](#_Toc88647443)

[4.27.18 拼帧后音频数据结构体 615](#_Toc88647444)

[4.27.19 视频解析数据结构体 616](#_Toc88647445)

[4.27.20 影像信息结构体 617](#_Toc88647446)

[4.27.21 矩形区域结构体 618](#_Toc88647447)

[4.27.22 录像查找信息结构体 619](#_Toc88647448)

[4.27.23 录像文件信息结构体 620](#_Toc88647449)

[4.27.24 按月查询录像分布信息结构体 621](#_Toc88647450)

[4.27.25 录像分布状态结构体 622](#_Toc88647451)

[4.27.26 通道列表结构体 623](#_Toc88647452)

[4.27.27 按时间回放录像参数结构体 624](#_Toc88647453)

[4.27.28 按文件名回放录像参数结构体 626](#_Toc88647454)

[4.27.29 拉框缩放结构体 628](#_Toc88647455)

[4.27.30 云台区域信息结构体 629](#_Toc88647456)

[4.27.31 云台守望信息结构体 630](#_Toc88647457)

[4.27.32 绝对坐标移动 631](#_Toc88647458)

[4.27.33 云台状态 631](#_Toc88647459)

[4.27.34 云台轨迹巡航路径结构体 632](#_Toc88647460)

[4.27.35 云台轨迹巡航路径列表 633](#_Toc88647461)

[4.27.36 云台轨迹巡航路径结构体 634](#_Toc88647462)

[4.27.37 云台轨迹巡航状态信息结构体 634](#_Toc88647463)

[4.27.38 所有云台预置位结构体 635](#_Toc88647464)

[4.27.39 云台预置位信息结构体 636](#_Toc88647465)

[4.27.40 云台预置位巡航路径详细信息结构体 636](#_Toc88647466)

[4.27.41 云台预置位巡航轨迹点结构体 637](#_Toc88647467)

[4.27.42 云台预置位巡航路径列表结构体 638](#_Toc88647468)

[4.27.43 云台方位标定信息结构体 639](#_Toc88647469)

[4.27.44 云台镜头当前视场角结构体 640](#_Toc88647470)

[4.27.45 云台经纬度信息单位 640](#_Toc88647471)

[4.27.46 日志查询条件结构体 641](#_Toc88647472)

[4.27.47 日志信息结构体 642](#_Toc88647473)

[4.27.48 告警信息查找条件结构体 644](#_Toc88647474)

[4.27.49 告警信息结构体 645](#_Toc88647475)

[4.27.50 告警日志查询条件列表 645](#_Toc88647476)

[4.27.51 告警日志查询条件结构体 646](#_Toc88647477)

[4.27.52 告警日志信息结构体 647](#_Toc88647478)

[4.27.53 人员信息列表结构体 649](#_Toc88647479)

[4.27.54 人员信息结构体 650](#_Toc88647480)

[4.27.55 成员地区信息结构体 653](#_Toc88647481)

[4.27.56 时间模板相关信息结构体 654](#_Toc88647482)

[4.27.57 成员证件信息结构体 654](#_Toc88647483)

[4.27.58 人脸图片信息列表结构体 655](#_Toc88647484)

[4.27.59 文件信息结构体 656](#_Toc88647485)

[4.27.60 自定义属性信息结构体 657](#_Toc88647486)

[4.27.61 员工信息结构体 658](#_Toc88647487)

[4.27.62 访客信息结构体 658](#_Toc88647488)

[4.27.63 人员信息结果列表结构体 659](#_Toc88647489)

[4.27.64 人员信息执行结果结构体 660](#_Toc88647490)

[4.27.65 人脸信息结果结构体 661](#_Toc88647491)

[4.27.66 布控任务信息结构体 661](#_Toc88647492)

[4.27.67 布控任务配置信息结构体 662](#_Toc88647493)

[4.27.68 人脸/车辆成员信息列表结构体 664](#_Toc88647494)

[4.27.69 告警联动配置信息结构体 665](#_Toc88647495)

[4.27.70 布控任务联动动作列表结构体 666](#_Toc88647496)

[4.27.71 联动动作列表信息结构体 666](#_Toc88647497)

[4.27.72 通道联动信息结构体 667](#_Toc88647498)

[4.27.73 联动使能参数结构体 668](#_Toc88647499)

[4.27.74 联动云台预置位列表结构体 668](#_Toc88647500)

[4.27.75 联动云台预置位信息结构体 669](#_Toc88647501)

[4.27.76 联动开关量输出结构体 670](#_Toc88647502)

[4.27.77 输出开关量的逻辑报警状态(手动告警)结构体 670](#_Toc88647503)

[4.27.78 计划（周）配置结构体 671](#_Toc88647504)

[4.27.79 计划（天）配置结构体 672](#_Toc88647505)

[4.27.80 时间段配置结构体 672](#_Toc88647506)

[4.27.81 布防信息结构体 673](#_Toc88647507)

[4.27.82 添加布控返回的布控信息列表结构体 674](#_Toc88647508)

[4.27.83 添加布控返回的布控信息结构体 674](#_Toc88647509)

[4.27.84 人员库信息结构体 675](#_Toc88647510)

[4.27.85 人员库信息列表结构体 676](#_Toc88647511)

[4.27.86 删除库标志位结构体 677](#_Toc88647512)

[4.27.87 所有人员库的容量信息结构体 677](#_Toc88647513)

[4.27.88 人脸库容量信息结构体 678](#_Toc88647514)

[4.27.89 人员信息查询条件结构体 679](#_Toc88647515)

[4.27.90 批量查询返回的基本信息结构体 680](#_Toc88647516)

[4.27.91 删除人脸布控返回的信息列表结构体 681](#_Toc88647517)

[4.27.92 批量操作信息结构体 681](#_Toc88647518)

[4.27.93 告警抓图URL查找条件 682](#_Toc88647519)

[4.27.94 告警抓拍图片信息 683](#_Toc88647520)

[4.27.95 抓拍图片信息结构体 684](#_Toc88647521)

[4.27.96 点坐标结构体 685](#_Toc88647522)

[4.27.97 手动录像结构体 686](#_Toc88647523)

[4.27.98 设备基本信息 686](#_Toc88647524)

[4.27.99 NTP参数 688](#_Toc88647525)

[4.27.100 地址结构体 688](#_Toc88647526)

[4.27.101 NTP列表 689](#_Toc88647527)

[4.27.102 通道视频流信息 690](#_Toc88647528)

[4.27.103 视频流信息列表 691](#_Toc88647529)

[4.27.104 视频流信息 692](#_Toc88647530)

[4.27.105 视频编码参数信息 693](#_Toc88647531)

[4.27.106 视频制式能力 694](#_Toc88647532)

[4.27.107 通道OSD的基本属性信息 695](#_Toc88647533)

[4.27.108 OSD时间配置信息 696](#_Toc88647534)

[4.27.109 OSD字符叠加信息 697](#_Toc88647535)

[4.27.110 区域 698](#_Toc88647536)

[4.27.111 通道OSD配置列表 698](#_Toc88647537)

[4.27.112 通道OSD配置 699](#_Toc88647538)

[4.27.113 通道OSD内容样式 700](#_Toc88647539)

[4.27.114 所有告警开关量输入信息 701](#_Toc88647540)

[4.27.115 告警开关量输出信息 702](#_Toc88647541)

[4.27.116 触发开关量输入告警 703](#_Toc88647542)

[4.27.117 所有告警开关量输入信息 703](#_Toc88647543)

[4.27.118 告警开关量输入信息 704](#_Toc88647544)

[4.27.119 输出开关量的逻辑报警状态列表(手动告警) 705](#_Toc88647545)

[4.27.120 输出开关量的逻辑报警状态(手动告警) 706](#_Toc88647546)

[4.27.121 触发或清除输出开关量的逻辑报警状态结构体(手动告警) 706](#_Toc88647547)

[4.27.122 输入开关量告警配置信息列表 707](#_Toc88647548)

[4.27.123 输入开关量告警信息 708](#_Toc88647549)

[4.27.124 设备图像设置 709](#_Toc88647550)

[4.27.125 图像曝光参数 710](#_Toc88647551)

[4.27.126 光圈信息 712](#_Toc88647552)

[4.27.127 快门信息 713](#_Toc88647553)

[4.27.128 增益信息 714](#_Toc88647554)

[4.27.129 宽动态信息 715](#_Toc88647555)

[4.27.130 测光信息 716](#_Toc88647556)

[4.27.131 测光区域 717](#_Toc88647557)

[4.27.132 左上角区域 717](#_Toc88647558)

[4.27.133 右下角区域 718](#_Toc88647559)

[4.27.134 昼夜模式信息 719](#_Toc88647560)

[4.27.135 补光灯信息 720](#_Toc88647561)

[4.27.136 白平衡信息 722](#_Toc88647562)

[4.27.137 网络配置信息 723](#_Toc88647563)

[4.27.138 VPN客户端指定模式信息 724](#_Toc88647564)

[4.27.139 设备DNS信息 726](#_Toc88647565)

[4.27.140 DNS地址 727](#_Toc88647566)

[4.27.141 网卡信息 728](#_Toc88647567)

[4.27.142 网卡的详细信息 729](#_Toc88647568)

[4.27.143 网卡信息中IPV4信息 730](#_Toc88647569)

[4.27.144 IPV4详细地址信息 731](#_Toc88647570)

[4.27.145 网卡信息中IPV6信息 732](#_Toc88647571)

[4.27.146 IPV6详细地址信息 733](#_Toc88647572)

[4.27.147 端口信息 734](#_Toc88647573)

[4.27.148 隐私遮盖配置信息 734](#_Toc88647574)

[4.27.149 区域配置结构体定义 735](#_Toc88647575)

[4.27.150 遮挡检测分析信息 736](#_Toc88647576)

[4.27.151 运动检测分析信息 737](#_Toc88647577)

[4.27.152 越界检测分析信息 738](#_Toc88647578)

[4.27.153 区域信息 739](#_Toc88647579)

[4.27.154 入侵检测分析信息 740](#_Toc88647580)

[4.27.155 入侵检测区域信息 740](#_Toc88647581)

[4.27.156 入侵检测区域点坐标 742](#_Toc88647582)

[4.27.157 硬盘信息列表 742](#_Toc88647583)

[4.27.158 硬盘详细信息 743](#_Toc88647584)

[4.27.159 聚焦信息 744](#_Toc88647585)

[4.27.160 昼夜模式信息 745](#_Toc88647586)

[4.27.161 透雾信息 745](#_Toc88647587)

[4.27.162 指定通道的图像增强参数 746](#_Toc88647588)

[4.27.163 音频输入参数配置信息 747](#_Toc88647589)

[4.27.164 音频输入通道信息 749](#_Toc88647590)

[4.27.165 夏令时配置信息 750](#_Toc88647591)

[4.27.166 夏令时配置 751](#_Toc88647592)

[4.27.167 时间配置 751](#_Toc88647593)

[4.27.168 录像计划配置信息 752](#_Toc88647594)

[4.27.169 录像计划规则 754](#_Toc88647595)

[4.27.170 视频计划（周）配置 754](#_Toc88647596)

[4.27.171 视频计划（天）配置 755](#_Toc88647597)

[4.27.172 视频时间段配置 756](#_Toc88647598)

[4.27.173 视频输入通道的抓图信息 757](#_Toc88647599)

[4.27.174 视频源分辨率信息 758](#_Toc88647600)

[4.27.175 计划（周）配置 759](#_Toc88647601)

[4.27.176 计划（天）配置 759](#_Toc88647602)

[4.27.177 时间段配置 760](#_Toc88647603)

[4.27.178 开关量计划（周）配置 761](#_Toc88647604)

[4.27.179 录像状态信息列表 762](#_Toc88647605)

[4.27.180 录像状态信息 763](#_Toc88647606)

[4.27.181 串口信息 763](#_Toc88647607)

[4.27.182 串口配置信息 764](#_Toc88647608)

[4.27.183 串口参数 765](#_Toc88647609)

[4.27.184 随路音频状态信息列表 766](#_Toc88647610)

[4.27.185 随路音频状态信息 767](#_Toc88647611)

[4.27.186 视频丢失告警的配置信息 768](#_Toc88647612)

[4.27.187 布控任务联动动作列表 768](#_Toc88647613)

[4.27.188 布控任务联动动作 769](#_Toc88647614)

[4.27.189 通道联动 770](#_Toc88647615)

[4.27.190 使能联动参数 771](#_Toc88647616)

[4.27.191 联动云台预置位 771](#_Toc88647617)

[4.27.192 联动云台预置位信息 772](#_Toc88647618)

[4.27.193 联动开关量输出 773](#_Toc88647619)

[4.27.194 输出开关量的逻辑报警状态(手动告警) 774](#_Toc88647620)

[4.27.195 使能联动参数 774](#_Toc88647621)

[4.27.196 遮挡检测告警的配置信息 775](#_Toc88647622)

[4.27.197 温度检测黑体信息结构体 776](#_Toc88647623)

[4.27.198 温度检测矫正信息结构体 777](#_Toc88647624)

[4.27.199 温度检测单位信息结构体 778](#_Toc88647625)

[4.27.200 运动检测区域类型信息 778](#_Toc88647626)

[4.27.201 运动检测宏块区域信息 779](#_Toc88647627)

[4.27.202 运动检测矩形区域信息列表 780](#_Toc88647628)

[4.27.203 运动检测矩形区域信息 781](#_Toc88647629)

[4.27.204 矩形区域坐标信息 782](#_Toc88647630)

[4.27.205 开关量布控任务联动动作 783](#_Toc88647631)

[4.27.206 阵列状态信息 784](#_Toc88647632)

[4.27.207 磁盘信息列表 784](#_Toc88647633)

[4.27.208 硬盘信息 785](#_Toc88647634)

[4.27.209 存储容器信息列表 786](#_Toc88647635)

[4.27.210 存储容器信息 788](#_Toc88647636)

[4.27.211 扩展存储容器信息 790](#_Toc88647637)

[4.27.212 硬盘Smart信息 792](#_Toc88647638)

[4.27.213 硬盘Smart检测信息 793](#_Toc88647639)

[4.27.214 照片接收服务器配置信息列表 794](#_Toc88647640)

[4.27.215 功能增强配置信息 795](#_Toc88647641)

[4.27.216 智能服务器配置列表 796](#_Toc88647642)

[4.27.217 智能服务器配置信息 797](#_Toc88647643)

[4.27.218 服务器用户信息 798](#_Toc88647644)

[4.27.219 运动检测告警参数 799](#_Toc88647645)

[4.27.220 管理服务器配置列表 800](#_Toc88647646)

[4.27.221 可视域参数信息 801](#_Toc88647647)

[4.27.222 验证信息 802](#_Toc88647648)

[4.27.223 存储配额信息 803](#_Toc88647649)

[4.27.224 自定义协议信息 804](#_Toc88647650)

[4.27.225 自定义协议配置信息 806](#_Toc88647651)

[4.27.226 自定义流协议信息 806](#_Toc88647652)

[4.27.227 自定义协议配置单例信息列表 807](#_Toc88647653)

[4.27.228 自定义协议配置信息 808](#_Toc88647654)

[4.27.229 存储策略信息 809](#_Toc88647655)

[4.27.230 视频存储策略信息 810](#_Toc88647656)

[4.27.231 图片存储策略信息 811](#_Toc88647657)

[4.27.232 后焦调节参数信息 812](#_Toc88647658)

[4.27.233 智能属性配置 812](#_Toc88647659)

[4.27.234 人脸属性配置 813](#_Toc88647660)

[4.27.235 温度属性配置 814](#_Toc88647661)

[4.27.236 图片矫正参数 815](#_Toc88647662)

[4.27.237 设备时间同步方式 816](#_Toc88647663)

[4.27.238 场景自动切换开关信息 817](#_Toc88647664)

[4.27.239 场景模板信息 817](#_Toc88647665)

[4.27.240 所有场景信息 818](#_Toc88647666)

[4.27.241 场景信息 819](#_Toc88647667)

[4.27.242 场景自动切换触发条件详细信息 820](#_Toc88647668)

[4.27.243 环境参数信息 821](#_Toc88647669)

[4.27.244 属性采集信息 822](#_Toc88647670)

[4.27.245 人脸属性开关信息 823](#_Toc88647671)

[4.27.246 行人属性开关信息 824](#_Toc88647672)

[4.27.247 非机动车属性开关信息 826](#_Toc88647673)

[4.27.248 车辆属性开关信息 827](#_Toc88647674)

[4.27.249 时间配置 830](#_Toc88647675)

[4.27.250 时间参数 831](#_Toc88647676)

[4.27.251 系统IP地址信息 832](#_Toc88647677)

[4.27.252 相对于电视墙的位置信息 833](#_Toc88647678)

[4.27.253 坐标信息 834](#_Toc88647679)

[4.27.254 硬盘Smart详细信息 834](#_Toc88647680)

[4.27.255 照片接收服务器配置信息 836](#_Toc88647681)

[4.27.256 智能服务器配置信息 837](#_Toc88647682)

[4.27.257 通道OSD所有内容 839](#_Toc88647683)

[4.27.258 通道OSD内容信息 839](#_Toc88647684)

[4.27.259 内容信息 840](#_Toc88647685)

[4.27.260 温度检测告警的配置信息 841](#_Toc88647686)

[4.27.261 温度检测基本信息结构体 842](#_Toc88647687)

[4.27.262 温度检测通用信息列表 843](#_Toc88647688)

[4.27.263 温度检测通用信息结构体 844](#_Toc88647689)

[4.27.264 温度检测比较信息列表 845](#_Toc88647690)

[4.27.265 温度检测比较信息结构体 846](#_Toc88647691)

[4.27.266 设备基本信息 848](#_Toc88647692)

[4.27.267 IP地址信息 851](#_Toc88647693)

[4.27.268 用户账号信息 852](#_Toc88647694)

[4.27.269 Onvif信息 853](#_Toc88647695)

[4.27.270 国标信息 853](#_Toc88647696)

[4.27.271 智能锁信息 854](#_Toc88647697)

[4.27.272 播放盒信息 855](#_Toc88647698)

[4.27.273 分辨率信息 857](#_Toc88647699)

[4.27.274 设备通道信息 857](#_Toc88647700)

[4.27.275 设备XW本地信号源通道信息 858](#_Toc88647701)

[4.27.276 设备通道基本信息 859](#_Toc88647702)

[4.27.277 设备基础信息 860](#_Toc88647703)

[4.27.278 时间信息 863](#_Toc88647704)

[4.27.279 客户端版本信息列表 864](#_Toc88647705)

[4.27.280 客户端版本信息 864](#_Toc88647706)

[4.27.281 设备信息 865](#_Toc88647707)

[4.27.282 设备详细信息 866](#_Toc88647708)

[4.27.283 设备固件信息 867](#_Toc88647709)

[4.27.284 定位信息 868](#_Toc88647710)

[4.27.285 WIFISnifferMac数组信息 868](#_Toc88647711)

[4.27.286 WIFISnifferMac信息 869](#_Toc88647712)

[4.27.287 视频通道详细信息 870](#_Toc88647713)

[4.27.288 视频通道详细信息 871](#_Toc88647714)

[4.27.289 视频通道详细信息拓展 873](#_Toc88647715)

[4.27.290 告警/事件上报信息 875](#_Toc88647716)

[4.27.291 告警消息 876](#_Toc88647717)

[4.27.292 事件信息 879](#_Toc88647718)

[4.27.293 事件资源信息 880](#_Toc88647719)

[4.27.294 拉告警信息 881](#_Toc88647720)

[4.27.295 客流量统计 881](#_Toc88647721)

[4.27.296 客流量统计命令 882](#_Toc88647722)

[4.27.297 客流量统计 883](#_Toc88647723)

[4.27.298 客流量上报时间间隔 884](#_Toc88647724)

[4.27.299 多通道客流量统计命令 885](#_Toc88647725)

[4.27.300 客流量统计信息 886](#_Toc88647726)

[4.27.301 XW设备能力集信息结构体 887](#_Toc88647727)

[4.27.302 电视墙能力集结构体 889](#_Toc88647728)

[4.27.303 视频输出制式列表结构体 890](#_Toc88647729)

[4.27.304 窗口能力集结构体 891](#_Toc88647730)

[4.27.305 底图能力集结构体 892](#_Toc88647731)

[4.27.306 虚拟LED能力集结构体 893](#_Toc88647732)

[4.27.307 轮巡能力集结构体 894](#_Toc88647733)

[4.27.308 特性能力集结构体 895](#_Toc88647734)

[4.27.309 XW设备支持的分屏信息结构体 897](#_Toc88647735)

[4.27.310 "XW"设备扩展视频输出制式能力集结构体 898](#_Toc88647736)

[4.27.311 通道数量信息结构体 898](#_Toc88647737)

[4.27.312 解码器通道信息结构体 899](#_Toc88647738)

[4.27.313 视频输入通道信息结构体 901](#_Toc88647739)

[4.27.314 XW视频输出通道信息结构体 902](#_Toc88647740)

[4.27.315 音频输入通道信息结构体 902](#_Toc88647741)

[4.27.316 音频输出通道信息结构体 903](#_Toc88647742)

[4.27.317 串口通道信息结构体 904](#_Toc88647743)

[4.27.318 电视墙信息结构体 905](#_Toc88647744)

[4.27.319 电视墙屏幕配置结构体 907](#_Toc88647745)

[4.27.320 特殊输出制式配置结构体 908](#_Toc88647746)

[4.27.321 特殊模组框配置结构体 908](#_Toc88647747)

[4.27.322 物理输出端口配置结构体 909](#_Toc88647748)

[4.27.323 虚拟LED信息结构体 910](#_Toc88647749)

[4.27.324 背景结构体 911](#_Toc88647750)

[4.27.325 文字信息描述结构体 912](#_Toc88647751)

[4.27.326 窗口信息结构体 914](#_Toc88647752)

[4.27.327 窗口分屏信息结构体 916](#_Toc88647753)

[4.27.328 窗口列表信息结构体 917](#_Toc88647754)

[4.27.329 批量开窗场景窗口信息结构体 918](#_Toc88647755)

[4.27.330 批量操作结果信息结构体 918](#_Toc88647756)

[4.27.331 批量开窗场景窗口返回信息结构体 919](#_Toc88647757)

[4.27.332 屏幕缩放信息结构体 920](#_Toc88647758)

[4.27.333 解码设备ID信息结构体 921](#_Toc88647759)

[4.27.334 视频源信息结构体 921](#_Toc88647760)

[4.27.335 鉴权信息结构体 923](#_Toc88647761)

[4.27.336 文字信息结构体 924](#_Toc88647762)

[4.27.337 被动解码发送控制参数信息结构体 925](#_Toc88647763)

[4.27.338 流业务的详细信息结构体 926](#_Toc88647764)

[4.27.339 轮巡资源基本信息结构体 927](#_Toc88647765)

[4.27.340 轮巡资源详细信息结构体 928](#_Toc88647766)

[4.27.341 轮巡视频源信息结构体 929](#_Toc88647767)

[4.27.342 远端实况的视频源基本信息结构体 930](#_Toc88647768)

[4.27.343 场景基本信息结构体 931](#_Toc88647769)

[4.27.344 场景修改信息结构体 932](#_Toc88647770)

[4.27.345 场景详细信息结构体 933](#_Toc88647771)

[4.27.346 所有底图信息结构体 935](#_Toc88647772)

[4.27.347 轮询资源列表信息结构体 936](#_Toc88647773)

[4.27.348 轮询资源信息结构体 936](#_Toc88647774)

[4.27.349 场景简要信息结构体 937](#_Toc88647775)

[4.27.350 轮巡资源绑定信息结构体 938](#_Toc88647776)

[4.27.351 轮巡播放控制信息结构体 939](#_Toc88647777)

[4.27.352 场景轮巡计划基本信息结构体 940](#_Toc88647778)

[4.27.353 场景计划定时切换模式信息结构体 941](#_Toc88647779)

[4.27.354 场景计划时间表模式信息结构体 942](#_Toc88647780)

[4.27.355 24小时时间表配置信息结构体 942](#_Toc88647781)

[4.27.356 场景轮巡计划播放控制信息结构体 943](#_Toc88647782)

[4.27.357 源端媒体流信息结构体 944](#_Toc88647783)

[4.27.358 目的端媒体流信息结构体 945](#_Toc88647784)

[4.27.359 媒体流具体信息结构体 946](#_Toc88647785)

[4.27.360 手动联动数据结构体 947](#_Toc88647786)

[4.27.361 矩形坐标结构体 947](#_Toc88647787)

[4.27.362 车牌识别报文数据信息结构体 948](#_Toc88647788)

[4.27.363 车牌识别报文XML解析信息结构体 949](#_Toc88647789)

[4.27.364 车牌识别报文图片解析信息结构体 950](#_Toc88647790)

[4.27.365 人脸识别记录结构体 951](#_Toc88647791)

[4.27.366 人脸抓拍告警记录比对信息结构体 952](#_Toc88647792)

[4.27.367 人脸库成员信息结构体 953](#_Toc88647793)

[4.27.368 抓拍图片信息结构体 955](#_Toc88647794)

[4.27.369 人脸库成员地区信息结构体 955](#_Toc88647795)

[4.27.370 成员证件信息结构体 956](#_Toc88647796)

[4.27.371 区域坐标结构体 957](#_Toc88647797)

[4.27.372 人脸抓拍信息结构体 958](#_Toc88647798)

[4.27.373 人脸位置信息结构体 960](#_Toc88647799)

[4.27.374 热度图报文数据信息结构体 960](#_Toc88647800)

[4.27.375 热度图报文XML解析信息结构体 961](#_Toc88647801)

[4.27.376 热度图报文区域解析信息结构体 962](#_Toc88647802)

[4.27.377 人员报警信息 963](#_Toc88647803)

[4.27.378 人脸通过记录信息 964](#_Toc88647804)

[4.27.379 人脸比对信息 965](#_Toc88647805)

[4.27.380 半结构化特征信息 967](#_Toc88647806)

[4.27.381 人脸属性信息 967](#_Toc88647807)

[4.27.382 人员属性 969](#_Toc88647808)

[4.27.383 结构化告警上报信息 970](#_Toc88647809)

[4.27.384 结构化数据信息 972](#_Toc88647810)

[4.27.385 目标信息 972](#_Toc88647811)

[4.27.386 人脸信息 973](#_Toc88647812)

[4.27.387 人员信息 975](#_Toc88647813)

[4.27.388 非机动车信息 976](#_Toc88647814)

[4.27.389 非机动车属性信息 978](#_Toc88647815)

[4.27.390 车辆信息 979](#_Toc88647816)

[4.27.391 车辆属性信息 980](#_Toc88647817)

[4.27.392 车牌属性信息 981](#_Toc88647818)

[4.27.393 车辆识别事件结构体 982](#_Toc88647819)

[4.27.394 车辆比对报警信息结构体 983](#_Toc88647820)

[4.27.395 车辆识别记录信息结构体 984](#_Toc88647821)

[4.27.396 车牌信息结构体 985](#_Toc88647822)

[4.27.397 订阅智能事件信息结构体 986](#_Toc88647823)

[4.27.398 智能事件信息结构体 987](#_Toc88647824)

[4.27.399 Lapi告警订阅信息结构体 988](#_Toc88647825)

[4.27.400 订阅信息成功返回信息结构体 989](#_Toc88647826)

[4.27.401 告警记录返回信息（人脸识别和车牌识别）结构体 990](#_Toc88647827)

[4.27.402 停车场出入口信息列表结构体 990](#_Toc88647828)

[4.27.403 停车场出入口信息结构体 991](#_Toc88647829)

[4.27.404 停车场信息列表结构体 992](#_Toc88647830)

[4.27.405 停车场信息结构体 993](#_Toc88647831)

[4.27.406 停车场出入口ID结构体 994](#_Toc88647832)

[4.27.407 车道信息列表结构体 995](#_Toc88647833)

[4.27.408 车道信息结构体 995](#_Toc88647834)

[4.27.409 自定义时间段结构体 997](#_Toc88647835)

[4.27.410 车辆放行信息结构体 997](#_Toc88647836)

[4.27.411 停车场支付订单信息结构体 998](#_Toc88647837)

[4.27.412 停车花费结构体 999](#_Toc88647838)

[4.27.413 停车场事件信息结构体 1000](#_Toc88647839)

[4.27.414 入场车辆抓拍信息结构体 1002](#_Toc88647840)

[4.27.415 出场车辆抓拍信息结构体 1002](#_Toc88647841)

[4.27.416 时间模板配置结构体 1003](#_Toc88647842)

[4.27.417 布控任务例外计划结构体 1005](#_Toc88647843)

[4.27.418 每天的布防计划具体信息结构体 1006](#_Toc88647844)

[4.27.419 通用ID列表结构体 1006](#_Toc88647845)

[4.27.420 时间模板结构体 1007](#_Toc88647846)

[4.27.421 门禁人员信息结构体 1008](#_Toc88647847)

[4.27.422 图片信息结构体 1010](#_Toc88647848)

[4.27.423 员工信息结构体 1011](#_Toc88647849)

[4.27.424 时间信息结构体 1011](#_Toc88647850)

[4.27.425 访客信息结构体 1012](#_Toc88647851)

[4.27.426 人员所持门禁卡信息结构体 1013](#_Toc88647852)

[4.27.427 黑名单信息结构体 1014](#_Toc88647853)

[4.27.428 人员列表结构体 1015](#_Toc88647854)

[4.27.429 授权信息结构体 1016](#_Toc88647855)

[4.27.430 人脸识别模块批量操作列表结构体 1017](#_Toc88647856)

[4.27.431 人脸识别模块批量操作信息结构体 1018](#_Toc88647857)

[4.27.432 出入记录信息结构体 1019](#_Toc88647858)

[4.27.433 人脸对比信息结构体 1021](#_Toc88647859)

[4.27.434 查询条件结构体 1021](#_Toc88647860)

[4.27.435 门禁人员基本信息结构体 1022](#_Toc88647861)

[4.27.436 访客记录信息结构体 1023](#_Toc88647862)

[4.27.437 门授权信息结构体 1025](#_Toc88647863)

[4.27.438 权限组信息结构体 1026](#_Toc88647864)

[4.27.439 人员核验结构体 1027](#_Toc88647865)

[4.27.440 人脸信息结构体 1029](#_Toc88647866)

[4.27.441 卡信息结构体 1030](#_Toc88647867)

[4.27.442 闸机信息结构体 1032](#_Toc88647868)

[4.27.443 库比对信息结构体 1033](#_Toc88647869)

[4.27.444 匹配人员信息结构体 1034](#_Toc88647870)

[4.27.445 图像相关信息结构体 1035](#_Toc88647871)

[4.27.446 车牌告警布控信息结构体 1036](#_Toc88647872)

[4.27.447 车辆成员信息 1037](#_Toc88647873)

[4.27.448 车辆信息 1038](#_Toc88647874)

[4.27.449 车牌信息 1038](#_Toc88647875)

[4.27.450 车辆信息列表结构体 1039](#_Toc88647876)

[4.27.451 批量操作成员列表结构体 1040](#_Toc88647877)

[4.27.452 告警记录返回信息结构体 1041](#_Toc88647878)

[4.27.453 车辆属性信息结构体 1041](#_Toc88647879)

[4.27.454 车牌告警布控信息结构体 1043](#_Toc88647880)

[4.27.455 建立透明通道参数结构体 1043](#_Toc88647881)

[4.27.456 用户信息结构体 1044](#_Toc88647882)

[4.27.457 通道权限信息结构体 1045](#_Toc88647883)

[4.27.458 修改用户详细信息结构体 1046](#_Toc88647884)

[4.27.459 用户密码信息结构体 1047](#_Toc88647885)

[4.27.460 用户信息列表结构体 1047](#_Toc88647886)

[4.27.461 组织信息结构体 1048](#_Toc88647887)

[4.27.462 待删除组织信息结构体 1049](#_Toc88647888)

[4.27.463 删除组织响应信息结构体 1050](#_Toc88647889)

[4.27.464 组织下通道信息结构体 1051](#_Toc88647890)

[4.27.465 组织通道信息结构体 1051](#_Toc88647891)

[4.27.466 网络端口号状态信息结构体 1052](#_Toc88647892)

[4.27.467 协议信息结构体 1053](#_Toc88647893)

[4.27.468 布控信息查询条件结构体 1054](#_Toc88647894)

[4.27.469 查找组织信息列表条件结构体 1054](#_Toc88647895)

[4.27.470 区域人数统计规则信息 1055](#_Toc88647896)

[4.27.471 检测区域规则信息 1056](#_Toc88647897)

[4.27.472 区域位置信息 1057](#_Toc88647898)

[4.27.473 绘制规则区域中每个顶点的坐标 1058](#_Toc88647899)

[4.27.474 绊线人数统计规则信息 1058](#_Toc88647900)

[4.27.475 检测绊线规则信息 1059](#_Toc88647901)

[4.27.476 绊线人数统计清零信息 1060](#_Toc88647902)

[4.27.477 联动球机信息列表 1061](#_Toc88647903)

[4.27.478 联动球机信息 1061](#_Toc88647904)

[4.27.479 系统时间配置 1062](#_Toc88647905)

[4.27.480 目标列表结构体 1063](#_Toc88647906)

[4.27.481 视图库信息结构体 1064](#_Toc88647907)

[4.27.482 温度信息结构体 1065](#_Toc88647908)

[4.27.483 录像时间信息 1066](#_Toc88647909)

[4.27.484 录像时间列表 1066](#_Toc88647910)

[4.27.485 检测区域图形定点坐标信息 1067](#_Toc88647911)

[4.27.486 规则信息 1068](#_Toc88647912)

[4.27.487 单个目标信息 1068](#_Toc88647913)

[4.27.488 音频输出参数 1069](#_Toc88647914)

[4.27.489 音频文件信息 1070](#_Toc88647915)

[4.27.490 通道信息列表 1071](#_Toc88647916)

[4.27.491 条件查询通道信息结构体 1072](#_Toc88647917)

[4.27.492 图片查询条件 1073](#_Toc88647918)

[4.27.493 图片查询结果 1073](#_Toc88647919)

[4.27.494 目标结果信息 1074](#_Toc88647920)

[4.27.495 目标信息列表 1075](#_Toc88647921)

[4.27.496 告警关联数据 1076](#_Toc88647922)

[4.27.497 图片告警基础信息 1077](#_Toc88647923)

[4.27.498 告警图片数据 1078](#_Toc88647924)

[4.27.499 人数统计告警信息 1079](#_Toc88647925)

[4.27.500 人数统计区域规则统计数据信息 1079](#_Toc88647926)

[4.27.501 人数统计绊线规则统计数据信息 1081](#_Toc88647927)

[4.27.502 人员密度统计信息 1082](#_Toc88647928)

[4.27.503 区域规则统计数据 1083](#_Toc88647929)

[4.27.504 伴线规则统计数据 1083](#_Toc88647930)

[4.27.505 人员密度统计数据 1084](#_Toc88647931)

[4.27.506 人员密度统计汇总信息 1085](#_Toc88647932)

[4.27.507 人员密度报警统计组信息 1086](#_Toc88647933)

[4.27.508 通道规则信息 1087](#_Toc88647934)

[4.27.509 滞留规则信息 1088](#_Toc88647935)

[4.27.510 报警规则信息 1089](#_Toc88647936)

[4.28 枚举定义 1089](#_Toc88647937)

[4.28.1 接入协议枚举 1089](#_Toc88647938)

[4.28.2 异常回调的消息类型枚举 1090](#_Toc88647939)

[4.28.3 设备类型枚举 1091](#_Toc88647940)

[4.28.4 码流类型枚举 1092](#_Toc88647941)

[4.28.5 媒体传输协议枚举 1092](#_Toc88647942)

[4.28.6 图像播放流畅性枚举 1092](#_Toc88647943)

[4.28.7 流模式枚举 1093](#_Toc88647944)

[4.28.8 传输类型枚举 1093](#_Toc88647945)

[4.28.9 起流协议枚举 1093](#_Toc88647946)

[4.28.10 媒体文件格式枚举 1094](#_Toc88647947)

[4.28.11 抓图图片格式枚举 1096](#_Toc88647948)

[4.28.12 音频格式枚举 1096](#_Toc88647949)

[4.28.13 音频位宽枚举 1097](#_Toc88647950)

[4.28.14 媒体数据流格式枚举 1097](#_Toc88647951)

[4.28.15 音频编码格式枚举 1097](#_Toc88647952)

[4.28.16 视频帧类型枚举 1098](#_Toc88647953)

[4.28.17 视频编码格式枚举 1098](#_Toc88647954)

[4.28.18 元数据显示类型枚举 1099](#_Toc88647955)

[4.28.19 视频显示比例枚举 1099](#_Toc88647956)

[4.28.20 录像存储类型枚举 1099](#_Toc88647957)

[4.28.21 录像存储位置枚举 1101](#_Toc88647958)

[4.28.22 录像查询位置枚举 1101](#_Toc88647959)

[4.28.23 录像状态枚举 1102](#_Toc88647960)

[4.28.24 录像下载速度枚举 1102](#_Toc88647961)

[4.28.25 播放/下载速度枚举 1103](#_Toc88647962)

[4.28.26 云台命令枚举 1105](#_Toc88647963)

[4.28.27 云台守望配置命令 1107](#_Toc88647964)

[4.28.28 云台守望模式 1107](#_Toc88647965)

[4.28.29 云台看守位操作命令 1108](#_Toc88647966)

[4.28.30 聚焦状态 1108](#_Toc88647967)

[4.28.31 巡航状态枚举 1108](#_Toc88647968)

[4.28.32 云台巡航操作枚举 1109](#_Toc88647969)

[4.28.33 云台预置位操作命令枚举 1109](#_Toc88647970)

[4.28.34 云台巡航操作枚举 1110](#_Toc88647971)

[4.28.35 云台方位枚举 1110](#_Toc88647972)

[4.28.36 回放控制命令枚举 1111](#_Toc88647973)

[4.28.37 日志主类型枚举 1111](#_Toc88647974)

[4.28.38 日志子类型枚举 1117](#_Toc88647975)

[4.28.39 告警信息查询类型枚举 1129](#_Toc88647976)

[4.28.40 告警日志查询条件枚举 1132](#_Toc88647977)

[4.28.41 告警类型枚举 1136](#_Toc88647978)

[4.28.42 人脸查找类型枚举 1146](#_Toc88647979)

[4.28.43 组织类型枚举 1147](#_Toc88647980)

[4.28.44 建模状态枚举 1147](#_Toc88647981)

[4.28.45 是否戴口罩枚举 1148](#_Toc88647982)

[4.28.46 查询条件逻辑类型枚举 1148](#_Toc88647983)

[4.28.47 告警子类型枚举 1149](#_Toc88647984)

[4.28.48 告警所属类型枚举 1170](#_Toc88647985)

[4.28.49 成员性别枚举 1170](#_Toc88647986)

[4.28.50 证件类型枚举 1171](#_Toc88647987)

[4.28.51 文件信息枚举 1171](#_Toc88647988)

[4.28.52 人脸处理结果状态码枚举 1172](#_Toc88647989)

[4.28.53 告警联动配置信息枚举 1174](#_Toc88647990)

[4.28.54 使能联动参数枚举 1174](#_Toc88647991)

[4.28.55 开关量状态枚举 1176](#_Toc88647992)

[4.28.56 布防计划使能枚举 1176](#_Toc88647993)

[4.28.57 星期枚举 1176](#_Toc88647994)

[4.28.58 告警布防类型枚举 1177](#_Toc88647995)

[4.28.59 人员库类型枚举 1178](#_Toc88647996)

[4.28.60 人脸布控操作结果错误码枚举 1178](#_Toc88647997)

[4.28.61 告警抓图类型枚举 1180](#_Toc88647998)

[4.28.62 5ePTZ显示模式枚举 1182](#_Toc88647999)

[4.28.63 鱼眼设备矫正模式枚举 1182](#_Toc88648000)

[4.28.64 设备安装模式枚举 1183](#_Toc88648001)

[4.28.65 鼠标移动模式枚举 1183](#_Toc88648002)

[4.28.66 录像类型枚举 1184](#_Toc88648003)

[4.28.67 录像状态枚举 1184](#_Toc88648004)

[4.28.68 设备配置命令 1184](#_Toc88648005)

[4.28.69 IP协议类型 1202](#_Toc88648006)

[4.28.70 视频图像质量 1203](#_Toc88648007)

[4.28.71 主码流类型 1203](#_Toc88648008)

[4.28.72 码率模式类型 1204](#_Toc88648009)

[4.28.73 GOP类型 1204](#_Toc88648010)

[4.28.74 图像扩展编码模式类型 1204](#_Toc88648011)

[4.28.75 OSD时间格式能力集枚举 1205](#_Toc88648012)

[4.28.76 OSD类型 1205](#_Toc88648013)

[4.28.77 OSD字体形式枚举 1205](#_Toc88648014)

[4.28.78 OSD字体大小枚举 1206](#_Toc88648015)

[4.28.79 日期格式 1206](#_Toc88648016)

[4.28.80 OSD时间格式枚举 1207](#_Toc88648017)

[4.28.81 OSD区域内对齐枚举 1208](#_Toc88648018)

[4.28.82 OSD边缘空的字符大小枚举 1208](#_Toc88648019)

[4.28.83 输入开关量运行模式 1208](#_Toc88648020)

[4.28.84 控制手动告警开关量状态命令 1209](#_Toc88648021)

[4.28.85 曝光模式 1209](#_Toc88648022)

[4.28.86 快门时间支持的取值 1210](#_Toc88648023)

[4.28.87 测光控制模式 1211](#_Toc88648024)

[4.28.88 昼夜模式类型 1212](#_Toc88648025)

[4.28.89 补光灯类型 1212](#_Toc88648026)

[4.28.90 白平衡模式 1213](#_Toc88648027)

[4.28.91 磁盘工作状态枚举 1213](#_Toc88648028)

[4.28.92 聚焦模式 1214](#_Toc88648029)

[4.28.93 昼夜模式 1214](#_Toc88648030)

[4.28.94 透雾模式枚举 1215](#_Toc88648031)

[4.28.95 音频采集口枚举 1215](#_Toc88648032)

[4.28.96 音频编码格式类型枚举 1215](#_Toc88648033)

[4.28.97 音频采样率枚举 1216](#_Toc88648034)

[4.28.98 音频输入模式枚举 1216](#_Toc88648035)

[4.28.99 夏令时偏移时间 1217](#_Toc88648036)

[4.28.100 星期 1217](#_Toc88648037)

[4.28.101 警前预录时间枚举 1218](#_Toc88648038)

[4.28.102 警后预录时间枚举定义 1218](#_Toc88648039)

[4.28.103 运动检测区域类型枚举 1219](#_Toc88648040)

[4.28.104 磁盘类型 1219](#_Toc88648041)

[4.28.105 磁盘工作模式 1220](#_Toc88648042)

[4.28.106 磁盘状态 1220](#_Toc88648043)

[4.28.107 存储容器状态 1220](#_Toc88648044)

[4.28.108 存储容器属性 1221](#_Toc88648045)

[4.28.109 地址类型 1221](#_Toc88648046)

[4.28.110 存储容器用途 1222](#_Toc88648047)

[4.28.111 硬盘健康评估状态 1222](#_Toc88648048)

[4.28.112 硬盘Smart检测状态 1223](#_Toc88648049)

[4.28.113 硬盘Smart检测类型 1223](#_Toc88648050)

[4.28.114 管理服务器接入协议类型 1224](#_Toc88648051)

[4.28.115 存储容量单位 1224](#_Toc88648052)

[4.28.116 传输协议 1224](#_Toc88648053)

[4.28.117 存储资源 1225](#_Toc88648054)

[4.28.118 满存储策略 1225](#_Toc88648055)

[4.28.119 实况业务流索引 1225](#_Toc88648056)

[4.28.120 图片存储模式 1226](#_Toc88648057)

[4.28.121 后焦控制操作命令 1226](#_Toc88648058)

[4.28.122 时区 1227](#_Toc88648059)

[4.28.123 补光灯支持的控制模式 1229](#_Toc88648060)

[4.28.124 硬盘Smart评估状态 1229](#_Toc88648061)

[4.28.125 照片服务器通信协议类型 1230](#_Toc88648062)

[4.28.126 照片服务器通信子协议类型 1231](#_Toc88648063)

[4.28.127 智能服务器接入协议类型 1231](#_Toc88648064)

[4.28.128 OSD内容类型 1232](#_Toc88648065)

[4.28.129 设备类型枚举定义 1233](#_Toc88648066)

[4.28.130 IP地址类型 1234](#_Toc88648067)

[4.28.131 锁信号 1234](#_Toc88648068)

[4.28.132 通道类型 1235](#_Toc88648069)

[4.28.133 编码通道状态 1236](#_Toc88648070)

[4.28.134 告警输入通道状态 1236](#_Toc88648071)

[4.28.135 告警输出通道状态 1236](#_Toc88648072)

[4.28.136 视频输入类型枚举 1237](#_Toc88648073)

[4.28.137 语言类型枚举 1237](#_Toc88648074)

[4.28.138 客户端类型 1240](#_Toc88648075)

[4.28.139 通道状态 1240](#_Toc88648076)

[4.28.140 通道类型 1241](#_Toc88648077)

[4.28.141 视频制式 1241](#_Toc88648078)

[4.28.142 POE口状态类型枚举 1241](#_Toc88648079)

[4.28.143 接入设备离线原因类型枚举 1242](#_Toc88648080)

[4.28.144 设备状态 1243](#_Toc88648081)

[4.28.145 设备子类型 1243](#_Toc88648082)

[4.28.146 接入协议 1245](#_Toc88648083)

[4.28.147 接入方式 1245](#_Toc88648084)

[4.28.148 通道下接入的设备类型(目前仅NVR使用) 1245](#_Toc88648085)

[4.28.149 告警/事件上报类型 1246](#_Toc88648086)

[4.28.150 告警资源类型 1246](#_Toc88648087)

[4.28.151 媒体类型枚举类型定义 1247](#_Toc88648088)

[4.28.152 解码层上报运行信息的类型的枚举定义 1248](#_Toc88648089)

[4.28.153 事件动作类型枚举 1249](#_Toc88648090)

[4.28.154 事件资源类型枚举 1250](#_Toc88648091)

[4.28.155 客流量统计模式类型 1253](#_Toc88648092)

[4.28.156 客流量统计报表类型 1253](#_Toc88648093)

[4.28.157 客流量报表类型 1254](#_Toc88648094)

[4.28.158 分辨率枚举 1254](#_Toc88648095)

[4.28.159 分屏能力枚举 1255](#_Toc88648096)

[4.28.160 窗口漫游能力枚举 1256](#_Toc88648097)

[4.28.161 开窗能力枚举 1256](#_Toc88648098)

[4.28.162 屏幕输出制式枚举 1257](#_Toc88648099)

[4.28.163 视频端口枚举 1259](#_Toc88648100)

[4.28.164 音频通道端口枚举 1259](#_Toc88648101)

[4.28.165 串口类型枚举 1260](#_Toc88648102)

[4.28.166 虚拟LED类型枚举 1260](#_Toc88648103)

[4.28.167 虚拟LED修改可选命令字枚举 1260](#_Toc88648104)

[4.28.168 虚拟LED字体枚举 1261](#_Toc88648105)

[4.28.169 虚拟LED字体大小枚举 1261](#_Toc88648106)

[4.28.170 虚拟LED字间距枚举 1262](#_Toc88648107)

[4.28.171 文字水平对齐方式枚举 1263](#_Toc88648108)

[4.28.172 文字垂直对齐方式枚举 1263](#_Toc88648109)

[4.28.173 文字滚动方式枚举 1263](#_Toc88648110)

[4.28.174 窗口放大类型枚举 1264](#_Toc88648111)

[4.28.175 窗口类型枚举 1264](#_Toc88648112)

[4.28.176 缩放状态枚举 1265](#_Toc88648113)

[4.28.177 分屏业务类型枚举 1265](#_Toc88648114)

[4.28.178 屏幕放大类型枚举 1265](#_Toc88648115)

[4.28.179 屏幕放大模式枚举 1266](#_Toc88648116)

[4.28.180 媒体传输协议枚举 1266](#_Toc88648117)

[4.28.181 实况业务类型枚举 1266](#_Toc88648118)

[4.28.182 解码类型枚举 1267](#_Toc88648119)

[4.28.183 加密类型枚举 1267](#_Toc88648120)

[4.28.184 文字位置枚举 1268](#_Toc88648121)

[4.28.185 资源管理类型枚举 1268](#_Toc88648122)

[4.28.186 流类型枚举 1268](#_Toc88648123)

[4.28.187 场景修改类型枚举 1269](#_Toc88648124)

[4.28.188 场景业务模式枚举 1269](#_Toc88648125)

[4.28.189 底图类型枚举 1270](#_Toc88648126)

[4.28.190 轮巡状态枚举 1270](#_Toc88648127)

[4.28.191 轮巡播放控制枚举 1270](#_Toc88648128)

[4.28.192 场景轮巡计划模式枚举 1271](#_Toc88648129)

[4.28.193 编码格式枚举 1271](#_Toc88648130)

[4.28.194 场景窗口修改可选命令字枚举 1272](#_Toc88648131)

[4.28.195 轮巡状态枚举 1273](#_Toc88648132)

[4.28.196 流业务类型枚举 1273](#_Toc88648133)

[4.28.197 门禁人员管理命令枚举 1274](#_Toc88648134)

[4.28.198 报警点通道控制命令枚举 1274](#_Toc88648135)

[4.28.199 手动联动类型枚举 1274](#_Toc88648136)

[4.28.200 人脸成员证件类型枚举 1275](#_Toc88648137)

[4.28.201 图片类型枚举 1275](#_Toc88648138)

[4.28.202 图片格式枚举 1276](#_Toc88648139)

[4.28.203 人脸通行记录类型枚举 1276](#_Toc88648140)

[4.28.204 年龄段枚举 1276](#_Toc88648141)

[4.28.205 是否戴眼镜标志枚举 1277](#_Toc88648142)

[4.28.206 眼镜款式枚举 1277](#_Toc88648143)

[4.28.207 上衣长短款式枚举 1278](#_Toc88648144)

[4.28.208 衣服颜色枚举 1278](#_Toc88648145)

[4.28.209 下衣长短款式枚举 1279](#_Toc88648146)

[4.28.210 身体抓拍朝向枚举 1280](#_Toc88648147)

[4.28.211 鞋子长短款式枚举 1280](#_Toc88648148)

[4.28.212 发型长短枚举 1281](#_Toc88648149)

[4.28.213 是否携包标志枚举 1281](#_Toc88648150)

[4.28.214 是否戴口罩枚举 1281](#_Toc88648151)

[4.28.215 上衣纹理枚举 1282](#_Toc88648152)

[4.28.216 人员运动方向枚举 1282](#_Toc88648153)

[4.28.217 结构化场景中非机动车速度类型枚举 1283](#_Toc88648154)

[4.28.218 结构化场景中非机动车相对画面运动方向枚举 1283](#_Toc88648155)

[4.28.219 非机动车类型枚举 1284](#_Toc88648156)

[4.28.220 车辆类型枚举 1285](#_Toc88648157)

[4.28.221 车牌颜色枚举 1286](#_Toc88648158)

[4.28.222 车牌类型枚举 1288](#_Toc88648159)

[4.28.223 通知类型枚举 1289](#_Toc88648160)

[4.28.224 智能告警类型枚举 1289](#_Toc88648161)

[4.28.225 LAPI协议订阅类型枚举 1290](#_Toc88648162)

[4.28.226 车道类型枚举 1290](#_Toc88648163)

[4.28.227 车辆放行类型枚举 1291](#_Toc88648164)

[4.28.228 订单支付状态枚举 1291](#_Toc88648165)

[4.28.229 支付类型枚举 1291](#_Toc88648166)

[4.28.230 人员类型枚举 1292](#_Toc88648167)

[4.28.231 访客状态枚举 1292](#_Toc88648168)

[4.28.232 采集来源枚举 1293](#_Toc88648169)

[4.28.233 匹配状态枚举 1293](#_Toc88648170)

[4.28.234 图像格式枚举 1294](#_Toc88648171)

[4.28.235 车牌颜色 1294](#_Toc88648172)

[4.28.236 车牌类型 1295](#_Toc88648173)

[4.28.237 车辆类型 1297](#_Toc88648174)

[4.28.238 结构化场景中非机动车速度类型枚举 1298](#_Toc88648175)

[4.28.239 结构化场景中非机动车相对画面运动方向枚举 1299](#_Toc88648176)

[4.28.240 通道权限枚举 1300](#_Toc88648177)

[4.28.241 响应状态类型枚举 1300](#_Toc88648178)

[4.28.242 NAT类型枚举 1300](#_Toc88648179)

[4.28.243 协议类型枚举 1301](#_Toc88648180)

[4.28.244 组织查找模式枚举 1301](#_Toc88648181)

[4.28.245 基本权限枚举 1302](#_Toc88648182)

[4.28.246 硬盘模式 1302](#_Toc88648183)

[4.28.247 能力集命令 1303](#_Toc88648184)

[4.28.248 OSD日期格式能力集 1304](#_Toc88648185)

[4.28.249 时间同步方式枚举 1304](#_Toc88648186)

[4.28.250 码流数据回调函数 1305](#_Toc88648187)

[4.28.251 人数统计类型 1305](#_Toc88648188)

[4.28.252 监听类型枚举 1306](#_Toc88648189)

[4.28.253 告警支持类型枚举 1306](#_Toc88648190)

[4.28.254 号牌颜色类型枚举 1307](#_Toc88648191)

[4.28.255 按位查询录像类型枚举 1308](#_Toc88648192)

[4.28.256 肤色枚举 1309](#_Toc88648193)

[4.28.257 情绪情况 1310](#_Toc88648194)

[4.28.258 微笑标志 1311](#_Toc88648195)

[4.28.259 胡子标志 1311](#_Toc88648196)

[4.28.260 目标类型枚举 1311](#_Toc88648197)

[4.28.261 规则类型枚举 1312](#_Toc88648198)

[4.28.262 规则触发类型枚举 1312](#_Toc88648199)

[4.28.263 目标记录类型 1313](#_Toc88648200)

[4.28.264 人数统计告警类型 1313](#_Toc88648201)

[4.28.265 人员密度报警统计类型 1314](#_Toc88648202)

[4.29 错误码列表 1314](#_Toc88648203)

[4.29.1 通用错误码 1314](#_Toc88648204)

[4.29.2 流媒体错误码 1317](#_Toc88648205)

[4.29.3 云台模块错误码 1325](#_Toc88648206)

[4.29.4 回放模块错误码 1326](#_Toc88648207)

[4.29.5 存储模块错误码 1327](#_Toc88648208)

[4.29.6 用户管理模块错误码 1328](#_Toc88648209)

[4.29.7 设备管理模块错误码 1329](#_Toc88648210)

[4.29.8 组织管理模块错误码 1330](#_Toc88648211)

[4.29.9 电视墙管理模块错误码 1331](#_Toc88648212)

[4.29.10 网络相关错误码 1332](#_Toc88648213)

[5. 产品相关接口定义 1334](#_Toc88648214)

[5.1 通用接口 1334](#_Toc88648215)

[5.2 网络摄像机 1339](#_Toc88648216)

[5.3 网络硬盘录像机 1344](#_Toc88648217)

[5.4 一体机 1350](#_Toc88648218)

[6. 帮助 1358](#_Toc88648219)

[6.1 编程环境配置 1358](#_Toc88648220)

[6.1.1 Windows下SDK开发编译环境 1358](#_Toc88648221)

[6.1.2 Linux下SDK开发编译环境 1359](#_Toc88648222)

[6.1.3 Android开发编译环境 1360](#_Toc88648223)

[6.1.4 IOS开发编译环境 1361](#_Toc88648224)

[6.2 注意事项 1362](#_Toc88648225)

[6.3 常见问题解答 1362](#_Toc88648226)

# 内容简介

## 前言

非常感谢您使用我们公司的产品，我们将竭诚为您提供最好的服务。

本手册可能包含技术上不准确的地方或文字错误。

本手册的内容将做定期的更新，恕不另行通知；更新的内容将会在本手册的新版本中加入。

我们随时会改进或更新本手册中描述的产品或程序。

## 阅读对象

本文档的阅读对象为具有一定开发能力，了解C/C++ 、C#等语言的开发、维护或者管理人员，对SDK对接有一定的经验与了解。

## 概述

NetDEVSDK（俗称网络设备SDK）是一套基于私有网络通信协议开发的，为IPC、NVR、VMS等产品服务的配套模块，包含了监控业务管理和控制、设备管理、资源管理以及系统维护接口。用户可以使用这些接口实现各种远程访问和控制设备等功能软件的二次开发。

主要提供设备管理、实况、回放、告警、云台、参数配置等视频业务功能。

本手册将会详细的介绍网络设备SDK中接口功能、使用说明及各接口间的调用关系；支持C/C++、C#、Objective-C等语言对接。

当您拿到此开发手册时，您应该已经拿到了完整的SDK开发包，包含头文件，静态库、动态库、demo，开发文档。

## 名词解释

|  |  |
| --- | --- |
| 名词 | 描述 |
| 实况预览 | 预览相机实时画面 |
| 回放预览 | 调取存储上的录像，进行历史视频预览功能，可进行播放控制 |
| 码流回调 | 在实况/回放时，SDK作为服务端，SDK可通过设置码流回调函数来获取视频流数据 |
| 录像下载 | 将存储上的录像下载到本地 |
| 本地录像 | 可以将正在预览的实时画面或者回放画面进行本地录制，录制后的录像存储在本地路径 |
| 本地抓拍 | 可以将正在预览的实时画面或者回放画面进行抓拍，抓拍的图片存储在本地路径 |

## 系统要求

* **Windows 32位网络设备SDK**

Windows 7、Windows 8、Windows 10

* **Windows 64位网络设备SDK**

Windows 7、Windows 8、Windows 10

* **Linux 64位网络设备SDK**

GCC版本要求大于等于gcc4.4.7；

支持系统：CentOS 6.5及以上

* **Linux 32位网络设备SDK**

GCC版本要求大于等于gcc4.4.7；

支持系统：CentOS 6.5及以上

* **Android网络设备SDK**

支持系统：Android 5及以上

* **IOS网络设备SDK**

支持系统：IOS 8.0及以上

## 适用的产品

* **编解码设备**

**NVR：**NVR202系列、NVR204系列、NVR208系列、NVR301系列、NVR302系列、NVR304系列、NVR308系列、NVR-B100系列、NVR-B200系列、NVR-S200系列、NVR-S300系列、NVS-B100系列；

**一体机：**VMS-B200-A16、VMS-B180-A、VMS-B230系列产品、VMS-B260系列产品；

* **网络摄像机**

IPC-B2系列、IPC-B3系列、IPC-B5系列、IPC-B6系列、IPC-B8系列、IPC-S2系列、IPC-S3系列、IPC-S5系列、IPC-S6系列、IPC-E2系列、IPC-E3系列、IPC-E5系列、IPC-E6系列、IPC2C系列、IPC2D系列、HIC25系列、HIC26系列、HIC27系列、HIC28系列、HIC29系列、HIC35系列、HIC36系列、HIC37系列、HIC54系列、HIC56系列、HIC58系列、HIC66系列、IPC67系列、HIC68系列、HIC74系列、HIC76系列、HIC78系列、IPC81系列、HIC85系列、IPC86系列、HIC95系列、HIC98系列、TIC22系列、TIC25系列、TIC68系列、TIC76系列、TIC78系列、EXC21系列、EXC22系列、EXC23系列、EXC26系列、EXC27系列、EXC66系列、EXC75系列；

# 版本更新

## 版本更新说明V2.5

|  |  |  |
| --- | --- | --- |
| **更新版本** | **更新日期** | **更新记录** |
| 2.5.15.0 | 2021-07-06 | 修改已知bug |

## 版本更新说明V2.6

|  |  |  |
| --- | --- | --- |
| **更新版本** | **更新日期** | **更新记录** |
| 2.6.0.0 | 2021-11-20 | 1. 修改已知bug； 2. 支持人数统计； 3. 支持火点告警； 4. 客户需求合入主线； 5. 细化起流错误码； |

# 编程指导

## 编程导引

NetDEVSDK用于接入IPC、NVR、VMS提供的各种业务功能接口，用户遵循SDK编程基本框架，同时参考编程指导各章节提供的接口使用说明，可以开发出基于SDK的各种媒体业务(实况、回放及下载、云台控制、参数配置等)。

* **SDK功能接口文件**

NetDEVSDK.h：提供了SDK使用的宏、枚举值、数据结构、功能接口函数定义。

* **编程说明**

1、字符串格式

非英文字符需UTF-8编码。

2、时间描述

Unix 纪元（January 1 1970 00:00:00 GMT）起的当前时间的秒数。

3、错误获取

在调用本开发包提供的函数时，如果函数内部产生异常(比如建立实况不成功)，异常情况会返回FALSE。获取错误码调用[NETDEV\_GetLastError()](#_获取接口错误码_1)，详细错误码请参见[错误码列表](#_错误码列表_1)。

## SDK调用主要流程

### 流程图



其中虚线框的流程是可选部分，不会影响其他流程和模块的功能使用。按实现功能的不同可以分成多个模块，实现每个模块的功能时初始化SDK、用户注册设备、查询通道、注销设备和释放SDK资源部分流程是必不可少的。

* 设置日志路径[NETDEV\_SetLogPath](#_设置日志文件路径)可选。
* 初始化[NETDEV\_Init](#_SDK_初始化)：对整个SDK的初始化，内存预分配等操作。
* 设备登录：适用于本地局域网以及公网下设备的登录，设备可以登录的前提能在本地PC 或者手机浏览器中通过IP地址或者域名能访问设备。本地设备登录[NETDEV\_Login\_V30](#_用户设备登录)：实现用户的登录功能，登录成功后，返回的用户ID作为其他功能操作的唯一标识。
* 查询IPC/NVR通道信息：登录成功后需要通过[NETDEV\_QueryVideoChlDetailListEx](#_查询视频通道信息列表)查询通道信息。
* 查询VMS下设备及通道信息：查询一体机下设备列表的接口有: [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) 。根据设备ID查询通道信息的相关接口有: [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)。
* 异常模块：异常的功能是用于接收SDK在实况、回放、下载、等模块发生异常的信息，具体流程详见[异常模块流程](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/异常模块流程.htm)。
* 告警模块：主要功能为获取设备的实时推送和历史告警，具体流程详见[告警模块流程](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)。
* 实况模块：从前端设备获取实时码流、解码显示功能。支持实况数据的获取。具体流程详见[实况模块流程](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)。
* 回放下载模块：可以查询历史录像并进行远程回放或者下载设备的录像文件，并可以进行解码或者存储。具体流程详见[回放及下载模块流程](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/回放及下载模块流程.htm)。
* 参数配置模块：获取及设置设备的参数，主要包括视频编码参数等信息。具体流程详见[参数配置模块流程](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/参数配置模块流程.htm)。
* 语音对讲模块： 语音对讲以及语音广播操作。具体流程详见[语音模块流程](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/非APP语音模块流程.htm)。
* 维护模块： 恢复出厂、重启相关操作。具体流程详见[维护模块流程](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/维护模块流程.htm)。
* 云台模块： 用于对云台、预置位、巡航等功能控制。具体流程详见[云台模块流程](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/云台控制流程.htm) 。
* 智能模块： 可以查询人脸、车牌以及客流量等相关功能。具体流程详见[智能模块流程](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/人员告警上报流程.htm) 。
* 用户注销[NETDEV\_Logout](#_用户注销)：用户退出系统。
* 释放SDK资源[NETDEV\_Cleanup](#_SDK清理)：退出系统，需要调用此接口释放SDK资源。

### NVR或者IPC登录示例代码

|  |
| --- |
| 下面示例代码为NVR或者IPC设备登录流程的C++示例代码，仅供参考。 |
| #include <iostream>  #include "NetDEVSDK.h"  void main**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.122"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "123456"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};**//分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  **if** **(**TRUE **==** bRet**)**  **{**  printf**(**"QueryVideoChlDetailList Success\n"**);**  **}**  **else**  **{**  printf**(**"QueryVideoChlDetailList failed, error code: %d\n"**,** NETDEV\_GetLastError**());**  **}**  //实现其他业务，如实况、回放、配置等功能  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

### VMS登录示例代码

|  |
| --- |
| 下面示例代码为VMS设备登录流程的C++示例代码，仅供参考。 |
| #include <iostream>  #include "NetDEVSDK.h"  void main**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.108"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "123456"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_PRIVATE**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  INT32 nDeviceType**[]** **=** **{**NETDEV\_DTYPE\_MAIN\_ENCODE**,** NETDEV\_DTYPE\_MAIN\_BAYONET**};** //编码设备、卡口设备  **for** **(**INT32 i **=**0**;** i**<sizeof(**nDeviceType**)/sizeof(**INT32**);** i**++)**  **{**  //通过设备类型查询设备基本信息  LPVOID lpDevFindHandle **=** NETDEV\_FindDevList**(**lUserID**,** nDeviceType**[**i**]);**  **if(NULL** **==** lpDevFindHandle**)**  **{**  **continue;**  **}**  //逐个获取设备基本信息  NETDEV\_DEV\_BASIC\_INFO\_S stDevInfo **=** **{**0**};**  **while(**TRUE **==** NETDEV\_FindNextDevInfo**(**lpDevFindHandle**,** **&**stDevInfo**))**  **{**  //通过设备ID查询设备编码通道信息  LPVOID lpChnFindHandle **=** NETDEV\_FindDevChnList**(**lUserID**,** stDevInfo**.**dwDevID**,** NETDEV\_CHN\_TYPE\_ENCODE**);**  **if(NULL** **==** lpChnFindHandle**)**  **{**  **continue;**  **}**  **else**  **{**  //逐个获取设备编码通道信息  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**);** //通道名称字符串格式为UTF-8  **}**  //关闭查找设备编码通道信息，释放资源  NETDEV\_FindCloseDevChn**(**lpChnFindHandle**);**  **}**  **}**  //关闭查找设备信息，释放资源  NETDEV\_FindCloseDevInfo**(**lpDevFindHandle**);**  **}**  //实现其他业务，如实况、回放、语音等功能  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

## 参数配置模块流程

### 流程图



### 图像配置示例代码

|  |
| --- |
| 下面示例代码为图像配置流程的C++示例代码，仅供参考。 |
| #include <iostream>  #include "NetDEVSDK.h"  void main**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.153"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "123456"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  //图像配置信息  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**);**  **}**  **}**  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

## 回放及下载模块流程

### 流程图



### 录像查询及通过时间回放示例代码

|  |
| --- |
| 下面示例代码为录像查询及通过时间回放流程的C++示例代码，仅供参考。 |
| #include <iostream>  #include <time.h>  #include "Windows.h"  #include "NetDEVSDK.h"  void playBack**(**LPVOID lpUserID**,**INT32 dwChannelID**,** NETDEV\_FINDDATA\_S stVodFile**)**  **{**  HWND hWnd **=** GetConsoleWindow**();** //获取窗口句柄  //开始回放  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**);** //millisecond  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**);** //millisecond  **}**  //停止回放  NETDEV\_StopPlayBack**(**lpPlayHandle**);**  **}**  void main**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  //查找录像文件  NETDEV\_FILECOND\_S stFileCond **=** **{**0**};**  stFileCond**.**dwChannelID **=** 1**;**  time\_t tUTCTime **=** time**(NULL);**  stFileCond**.**tEndTime **=** time**(&**tUTCTime**);** //当前时间戳  stFileCond**.**tBeginTime **=** time**(&**tUTCTime**)-**60**;** //当前时间戳前60秒  LPVOID dwFileHandle **=** NETDEV\_FindFile**(**lUserID**,** **&**stFileCond**);**  **if(NULL** **==** dwFileHandle**)**  **{**  printf**(**"FindFile fail, error code: %d\n"**,** NETDEV\_GetLastError**());**  **}**  **else**  **{**  //逐个获取查找录像结果  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**);**  **}**  //停止查找  NETDEV\_FindClose**(**dwFileHandle**);**  **}**  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

### 录像查询及通过时间下载示例代码

|  |
| --- |
| 下面示例代码为录像查询及通过时间下载录像流程的C++示例代码，仅供参考。 |
| #include <iostream>  #include <time.h>  #include "Windows.h"  #include "NetDEVSDK.h"  void saveRecordFile**(**LPVOID lpUserID**,**NETDEV\_FINDDATA\_S stVodFile**)**  **{**  //开始下载录像  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**);** //millisecond  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**);** //millisecond  **}**  //停止下载录像  NETDEV\_StopGetFile**(**lpPlayHandle**);**  **}**  void main**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  //查找录像文件  NETDEV\_FILECOND\_S stFileCond **=** **{**0**};**  stFileCond**.**dwChannelID **=** 1**;**  time\_t tUTCTime **=** time**(NULL);**  stFileCond**.**tEndTime **=** time**(&**tUTCTime**);** //当前时间戳  stFileCond**.**tBeginTime **=** time**(&**tUTCTime**)-**60**;** //当前时间戳前60秒  LPVOID dwFileHandle **=** NETDEV\_FindFile**(**lUserID**,** **&**stFileCond**);**  **if(NULL** **==** dwFileHandle**)**  **{**  printf**(**"FindFile fail, error code: %d\n"**,** NETDEV\_GetLastError**());**  **}**  **else**  **{**  //逐个获取查找录像结果  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**);**  **}**  //停止查找  NETDEV\_FindClose**(**dwFileHandle**);**  **}**  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

### 录像查询及通过文件名下载示例代码

|  |
| --- |
| 下面示例代码为录像查询及通过文件名下载录像流程的C++示例代码，仅供参考。 |
| #include <iostream>  #include <time.h>  #include "Windows.h"  #include "NetDEVSDK.h"  void saveRecordFile**(**LPVOID lpUserID**,**NETDEV\_FINDDATA\_S stVodFile**)**  **{**  //开始下载录像  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**);** //millisecond  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**);** //millisecond  **}**  //停止下载录像  NETDEV\_StopGetFile**(**lpPlayHandle**);**  **}**  void main**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  //查找录像文件  NETDEV\_FILECOND\_S stFileCond **=** **{**0**};**  stFileCond**.**dwChannelID **=** 1**;**  time\_t tUTCTime **=** time**(NULL);**  stFileCond**.**tEndTime **=** time**(&**tUTCTime**);** //当前时间戳  stFileCond**.**tBeginTime **=** time**(&**tUTCTime**)-**60**;** //当前时间戳前60秒  LPVOID dwFileHandle **=** NETDEV\_FindFile**(**lUserID**,** **&**stFileCond**);**  **if(NULL** **==** dwFileHandle**)**  **{**  printf**(**"FindFile fail, error code: %d\n"**,** NETDEV\_GetLastError**());**  **}**  **else**  **{**  //逐个获取查找录像结果  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**);**  **}**  //停止查找  NETDEV\_FindClose**(**dwFileHandle**);**  **}**  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

## 实况模块流程

### 流程图



### 实况示例代码

|  |
| --- |
| 下面示例代码为实况流程的C++示例代码，仅供参考。 |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void main**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  //实况预览  HWND hWnd **=** GetConsoleWindow**();** //获取窗口句柄  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**);** //millisecond  //停止实况预览  NETDEV\_StopRealPlay**(**lpPlayHandle**);**  **}**    //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

## 语音模块流程

### 语音对讲模块流程

#### 流程图



#### 示例代码

|  |
| --- |
| 下面示例代码为语音对讲流程的C++示例代码，仅供参考。 |
| #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()  {  //初始化  NETDEV\_Init();  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout = {0};  stRevTimeout.dwRevTimeOut = 5;  stRevTimeout.dwFileReportTimeOut = 30;  NETDEV\_SetRevTimeOut(&stRevTimeout);  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo = {0};  strncpy(stDevLoginInfo.szIPAddr, "192.168.3.220", sizeof(stDevLoginInfo.szIPAddr)); //设备IP地址  strncpy(stDevLoginInfo.szUserName, "admin", sizeof(stDevLoginInfo.szUserName)); //设备登录用户名  strncpy(stDevLoginInfo.szPassword, "Admin12345", sizeof(stDevLoginInfo.szPassword)); //设备登录密码  stDevLoginInfo.dwPort = 80; //设备服务端口  stDevLoginInfo.dwLoginProto = NETDEV\_LOGIN\_PROTO\_ONVIF; //登录协议  //输出参数，仅私有协议登录有效  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());  //释放SDK资源  NETDEV\_Cleanup();  return;  }  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx[128] = {0}; //分配个通道  INT32 dwCount = 128; //128个通道  BOOL bRet = NETDEV\_QueryVideoChlDetailListEx(lUserID, &dwCount, stVideoChlDetailInfoEx);  if (FALSE == bRet && NETDEV\_E\_NEEDMOREDATA == NETDEV\_GetLastError())  {  //默认分配数组大小小于实际通道数，重新分配内存获取  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);  //释放分配内存  delete []pstVideoChlDetailInfoEx;  }  //开启语音对讲  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); //millisecond  //停止语音对讲  NETDEV\_StopVoiceCom(lpPlayHandle);  }    //注销用户  NETDEV\_Logout(lUserID);  //释放SDK资源  NETDEV\_Cleanup();  } |

### 语音转发模块流程

#### 流程图



#### 示例代码

|  |
| --- |
| 下面示例代码为语音转发流程的C++示例代码，仅供参考。 |
| #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**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  //开启语音对讲服务  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**};** //需上层获取到PCM格式音频数据  int dwBufLen **=** 640**;** //单次发送长度，160的倍数 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**);** //传输语音数据  Sleep**(**20000**);** //millisecond  //停止语音对讲服务  NETDEV\_StopInputVoiceSrv**(**lpPlayHandle**);**  **}**    //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

## 维护模块流程

### 流程图



### 示例代码

|  |
| --- |
| 下面示例代码为语音转发流程的C++示例代码，仅供参考。 |
| #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**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  //重启设备  bRet **=** NETDEV\_Reboot**(**lUserID**);**  **if** **(**TRUE **!=** bRet**)**  **{**  printf**(**"Reboot failed\n"**);**  **}**  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

## 异常模块流程

### 流程图



### 示例代码

|  |
| --- |
| 下面示例代码为异常处理流程的C++示例代码，仅供参考。 |
| #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**:** //预览时重连  printf**(**"----------Device offline--------\n"**);**  **break;**  **default:**  **break;**  **}**  **}**  void main**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //注册异常回调  NETDEV\_SetExceptionCallBack**(**ExceptionCallBack\_PF**,NULL);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  Sleep**(**120000**);** //millisecond  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

## 告警模块流程

### 流程图



### 示例代码

|  |
| --- |
| 下面示例代码为告警流程的C++示例代码，仅供参考。 |
| #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**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **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**);** //millisecond  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

## 云台模块流程

### 云台控制流程

#### 流程图



#### 示例代码

|  |
| --- |
| 下面示例代码为云台控制流程的C++示例代码，仅供参考。 |
| #include <iostream>  #include "Windows.h"  #include "NetDEVSDK.h"  void main**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  INT32 dwChannelID **=** 1**;**  //实况预览  HWND hWnd **=** GetConsoleWindow**();** //获取窗口句柄  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**);** //millisecond  NETDEV\_PTZControl\_Other**(**lUserID**,** dwChannelID**,** NETDEV\_PTZ\_TILTDOWN**,** 6**);**  Sleep**(**5000**);** //millisecond  NETDEV\_PTZControl\_Other**(**lUserID**,** dwChannelID**,** NETDEV\_PTZ\_PANRIGHT**,** 6**);**  Sleep**(**5000**);** //millisecond  NETDEV\_PTZControl\_Other**(**lUserID**,** dwChannelID**,** NETDEV\_PTZ\_PANLEFT**,** 6**);**  Sleep**(**5000**);** //millisecond  NETDEV\_PTZControl\_Other**(**lUserID**,** dwChannelID**,** NETDEV\_PTZ\_ALLSTOP**,** 6**);**  **if** **(NULL** **!=** lpPlayHandle**)**  **{**  //停止实况预览  NETDEV\_StopRealPlay**(**lpPlayHandle**);**  **}**  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

### 预置位和巡航模块流程

#### 流程图



## 智能模块流程

### 人员模块告警流程

#### 流程图



人员告警上报流程图



人员、人员库及布控操作流程图



人脸识别历史记录查询流程图

#### NVR或IPC人脸识匹配据上报示例代码

|  |
| --- |
| 下面示例代码为NVR或者IPC人脸匹配流程的C++示例代码，仅供参考。 |
| #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**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  //注册人脸匹配数据回调  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**);** //millisecond  bRet **=** NETDEV\_UnSubLapiAlarm**(**lUserID**,** stSubSuccInfo**.**udwID**);**  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

#### VMS人脸匹配数据上报示例代码

|  |
| --- |
| 下面示例代码为VMS人脸匹配流程的C++示例代码，仅供参考。 |
| #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**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.108"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "123456"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_PRIVATE**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //订阅告警，包含人脸匹配数据订阅  NETDEV\_SetAlarmCallBack\_V30**(**lUserID**,** AlarmMessCallBack\_PF**,** lUserID**);**  //注册人脸匹配数据回调  NETDEV\_SetPersonAlarmCallBack**(**lUserID**,**PersonAlarmCallBack**,**lUserID**);**  Sleep**(**120000**);** //millisecond  //取消订阅告警  NETDEV\_SetAlarmCallBack\_V30**(**lUserID**,** **NULL,** lUserID**);**  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

### 客流量模块功能流程

#### 流程图



### 车辆模块功能流程

#### 流程图



车辆告警流程图



车辆、车辆库及布控流程图



车辆识别历史记录查询流程图

#### NVR或IPC车辆匹配数据上报示例代码

|  |
| --- |
| 下面示例代码为NVR或者IPC车辆匹配流程的C++示例代码，仅供参考。 |
| #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**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  //注册车辆识别数据回调  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**);** //millisecond  bRet **=** NETDEV\_UnSubLapiAlarm**(**lUserID**,** stSubSuccInfo**.**udwID**);**  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

### 结构化数据（过人过车）流程

#### 流程图



#### NVR或IPC结构化数据上报示例代码

|  |
| --- |
| 下面示例代码为NVR或者IPC结构化数据流程的C++示例代码，仅供参考。 |
| #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**()**  **{**  //初始化  NETDEV\_Init**();**  //设置连接时间  NETDEV\_REV\_TIMEOUT\_S stRevTimeout **=** **{**0**};**  stRevTimeout**.**dwRevTimeOut **=** 5**;**  stRevTimeout**.**dwFileReportTimeOut **=** 30**;**  NETDEV\_SetRevTimeOut**(&**stRevTimeout**);**  //登录参数，包括设备地址、登录用户、密码等  NETDEV\_DEVICE\_LOGIN\_INFO\_S stDevLoginInfo **=** **{**0**};**  strncpy**(**stDevLoginInfo**.**szIPAddr**,** "192.168.3.220"**,** **sizeof(**stDevLoginInfo**.**szIPAddr**));** //设备IP地址  strncpy**(**stDevLoginInfo**.**szUserName**,** "admin"**,** **sizeof(**stDevLoginInfo**.**szUserName**));** //设备登录用户名  strncpy**(**stDevLoginInfo**.**szPassword**,** "Admin12345"**,** **sizeof(**stDevLoginInfo**.**szPassword**));** //设备登录密码  stDevLoginInfo**.**dwPort **=** 80**;** //设备服务端口  stDevLoginInfo**.**dwLoginProto **=** NETDEV\_LOGIN\_PROTO\_ONVIF**;** //登录协议  //输出参数，仅私有协议登录有效  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**());**  //释放SDK资源  NETDEV\_Cleanup**();**  **return;**  **}**  //通道查询  NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S stVideoChlDetailInfoEx**[**128**]** **=** **{**0**};** //分配128个通道  INT32 dwCount **=** 128**;** //128个通道  BOOL bRet **=** NETDEV\_QueryVideoChlDetailListEx**(**lUserID**,** **&**dwCount**,** stVideoChlDetailInfoEx**);**  **if** **(**FALSE **==** bRet **&&** NETDEV\_E\_NEEDMOREDATA **==** NETDEV\_GetLastError**())**  **{**  //默认分配数组大小小于实际通道数，重新分配内存获取  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**);**  //释放分配内存  **delete** **[]**pstVideoChlDetailInfoEx**;**  **}**  //注册结构化数据回调  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**);** //millisecond  bRet **=** NETDEV\_UnSubLapiAlarm**(**lUserID**,** stSubSuccInfo**.**udwID**);**  //注销用户  NETDEV\_Logout**(**lUserID**);**  //释放SDK资源  NETDEV\_Cleanup**();**  **}** |

# 基本接口定义

## SDK初始化

### SDK初始化

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_Init(); |

**接口描述：**

初始化SDK，调用其他SDK接口的前提。

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 程序里面只需调用1次[NETDEV\_Init](#_SDK_初始化)，退出时需要调用1次[NETDEV\_Cleanup](#_SDK清理)

**See Also：**

[NETDEV\_Cleanup](#_SDK清理)

### SDK清理

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ Cleanup (); |

**接口描述：**

SDK清理。

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[[NETDEV\_GetLastError](#_获取接口错误码_1)](#_获取接口错误码)，通过错误码判断出错原因。

**Remarks**：

* 在调用[NETDEV\_Cleanup](#_SDK清理)的时候，不能同时调用其他任何SDK接口；
* [NETDEV\_Init](#_SDK_初始化)和[NETDEV\_Cleanup](#_SDK清理)需要配对使用；
* 程序里面只需调用1次[NETDEV\_Init](#_SDK_初始化)，退出时需要调用1次[NETDEV\_Cleanup](#_SDK清理)；

**See Also：**

[NETDEV\_Init](#_SDK_初始化)

## SDK本地功能

### 设置日志大小和数量

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ConfigLogFile  (  INT32 dwLogFileSize,  INT32 dwLogFileNum  ); |

**接口描述：**

设置日志文件大小和数量；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| dwLogFileSize | IN | 单个日志文件大小，单位：字节； |
| dwLogFileNum | IN | 日志文件个数 |

**Return Values：**

TRUE表示成功,其他表示失败。

**Remarks**：

* 此接口可在[NETDEV\_Init](#_SDK初始化)接口之前调用；
* 默认日志数量为10个，单个大小为30M；

**See Also：**

[NETDEV\_SetLogPath](#_设置日志文件路径)、[NETDEV\_SetWriteLogFlag](#_设置写入日志的标记)

### 设置日志文件路径

**接口名称：**

|  |
| --- |
| BOOL    STDCALL   NETDEV\_SetLogPath  (  CHAR \*       pszLogPath  ); |

**接口描述：**

设置日志文件路径；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| pszLogPath | IN | 日志路径(不包含文件名)； |

**Return Values：**

TRUE表示成功,其他表示失败。

**Remarks**：

* 此接口可在[NETDEV\_Init](#_SDK初始化)接口之前调用；
* 默认日志路径为程序运行路径下；
* 设置日志路径时，日志文件所在路径必须为已存在路径；

**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)

### 设置写入日志的标记

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetWriteLogFlag  (  BOOL bWriteLogFlag  ); |

**接口描述：**

设置写入日志的标记；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| bWriteLogFlag | IN | 写入日志的标记 TRUE:写入日志,FALSE：不写入日志； |

**Return Values：**

TRUE表示成功,其他表示失败。

**Remarks**：

* 此接口可在[NETDEV\_Init](#_SDK初始化)接口之前调用；
* 写日志功能默认开启，写日志标记为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)

### 获取SDK版本信息

**接口名称：**

|  |
| --- |
| INT32 STDCALL NETDEV\_GetSDKVersion(); |

**接口描述：**

获取SDK的版本信息；

**Return Values：**

SDK版本信息。

**Remarks**：

* 在两个高字节中高8位表示主版本,低八位表示次版本.两个低字节表示附加版本号如0x01080000：表示版本为1.8.0.0；

### 获取接口错误码

**接口名称：**

|  |
| --- |
| INT32        STDCALL NETDEV\_GetLastError(); |

**接口描述：**

接口调用失败后，获取对应错误码；

**Return Values：**

返回值为错误码。错误码主要分为通用错误、媒体相关错误、云台模块错误、回放模块错误等错误码，详见[错误码列表](#_错误码列表_1)。

**Remarks**：

* 仅SDK接口返回失败时，才可调用该接口获取错误码；

### 注册异常回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetExceptionCallBack  (  NETDEV\_ExceptionCallBack\_PF cbExceptionCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册SDK接收异常.重连等消息的回调函数；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| cbExceptionCallBack | IN | 接收异常消息的回调函数,回调当前异常的相关信息 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄， [NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwType | IN | 异常或重连等消息的类型，详见枚举 [NETDEV\_EXCEPTION\_TYPE\_E](#_异常回调的消息类型枚举) |
| lpExpHandle | IN | 出现异常的相应类型的句柄 |
| lpUserData | IN | 用户数据，对应设置回调时传入的lpUserData |
| pstExceptionOutputInfo | IN | 异常输出信息，仅播放器上报异常有用到，参考 [NETDEV\_EXCEPTION\_OUTPUT\_INFO\_S](#_播放器异常输出信息) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 设置最大解码路数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetMaxDecChlNum  (  UINT32 dwMaxdecNum  ) ; |

**接口描述：**

设置最大解码路数；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| dwMaxdecNum | IN | 最大解码通道数； |

**Return Values：**

TRUE表示成功,其他表示失败。

**Remarks**：

* 此接口必须在[NETDEV\_Init](#_SDK初始化)接口之前调用；
* 该接口调用失败，表示解码路数为128路；
* SDK最多支持128路解码；
* 解码路数包括：实况路数、回放路数、录像下载路数的总和；

### 设置保活等待间隔及次数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetConnectTime  (  INT32 dwWaitTime,  INT32 dwTrytimes  ); |

**接口描述：**

设置保活参数；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| dwWaitTime | IN | 间隔等待时间，单位：秒； |
| dwTrytimes | IN | 尝试连接次数; |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 间隔等待时间及尝试连接次数必须大于0；

### 设置接收超时时间

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetRevTimeOut  (  [LPNETDEV\_REV\_TIMEOUT\_S](#_超时时间结构体) pstRevTimeout  ); |

**接口描述：**

设置SDK接口接收超时时间；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| pstRevTimeout | IN | 超时时间，详见：[NETDEV\_REV\_TIMEOUT\_S](#_超时时间结构体)； |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 网络消息超时时间必须大于等于1秒；
* 文件传输超时时间必须大于等于30秒；

### 设置本地监听地址

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetListenAddr  (  INT32 dwListenType,  CHAR\* pszAddr  ); |

**接口描述：**

设置本地监听地址

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| dwListenType | IN | 监听功能类型，参考枚举 [NETDEV\_LISTEN\_TYPE\_E](#_监听类型枚举) |
| pszAddr | IN | 监听地址 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 此接口可在[NETDEV\_Init](#_SDK初始化)接口之前调用；

## 设备搜索

### 设置设备搜索回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetDiscoveryCallBack  (  NETDEV\_DISCOVERY\_CALLBACK\_PF cbDiscoveryCallBack,  LPVOID lpUserData  ); |

**接口描述：**

设置设备搜索回调函数

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| cbDiscoveryCallBack | IN | 设备搜索回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| pstDevInfo | IN | 设备信息，详见结构体 [NETDEV\_DISCOVERY\_DEVINFO\_S](#_设备发现的设备信息结构体) |
| lpUserData | IN | 用户数据，对应设置回调时传入的lpUserData |

**Return Values：**

返回0表示失败，其他值表示返回的用户ID值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 此接口调用必须在[NETDEV\_Discovery](#_设备搜索)接口之前

**See Also：**

[NETDEV\_Discovery](#_设备搜索)

### 设备搜索

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_Discovery  (  CHAR \*pszBeginIP,  CHAR \*pszEndIP  ); |

**接口描述：**

设备发现

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| pszBeginIP | IN | 起始IP地址 |
| pszEndIP | IN | 结束IP地址 |

**Return Values：**

返回0表示失败，其他值表示返回的用户ID值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 若pszBeginIP和pszEndIP都是"0.0.0.0"，则表示自动搜索，否则为按网段搜索；
* 先注册设备发现相关的回调，再调用此接口发现设备，发现的设备信息在回调中反映；

**See Also：**

[NETDEV\_SetDiscoveryCallBack](#_设置设备搜索回调函数)

## 免登陆配置

### 修改设备IP地址

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyDeviceAddr([LPNETDEV\_DEV\_ADDR\_INFO\_S](#_设备地址信息结构体) pstDevAddrInfo); |

**接口描述：**

修改设备IP地址（无需登录）

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| pstDevAddrInfo | IN | 要修改的设备地址信息 |

**Return Values：**

返回0表示失败，其他值表示返回的用户ID值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 不登录修改设备IP地址；

**See Also：**

[NETDEV\_SetDiscoveryCallBack](#_设置设备搜索回调函数)、[NETDEV\_Discovery](#_设备搜索)

## 用户登录

### 用户设备登录

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

用户登录设备

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| pstDevLoginInfo | IN | 设备登录信息 |
| pstSELogInfo | OUT | 安全登录信息 |

**Return Values：**

返回0表示失败，其他值表示返回的用户ID值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 本地登录时，SDK使用端口与web页面访问设备一致，默认为80端口；
* 安全登录信息此字段仅适用于使用私有协议登录的设备，如：VMS；
* NVR/IPC仅支持ONVIF协议登录，VMS仅支持私有协议登录；

**See Also：**

[NETDEV\_Logout](#_用户注销)

### 用户注销

**接口名称：**

|  |
| --- |
| BOOL STDCALL     NETDEV\_Logout  (  LPVOID    lpUserID  ) ; |

**接口描述：**

用户注销

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄， [NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |

**Return Values：**

返回0表示失败，其他值表示返回的用户ID值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_Login\_V30](#_用户设备登录)

## 实时预览

### 实时预览

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

启动实时预览

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstPreviewInfo | IN | 预览参数，参考结构体 [NETDEV\_PREVIEWINFO\_S](#_实况预览参数结构体) |
| cbPlayDataCallBack | IN | 码流数据回调函数 |
| lpUserData | IN | 用户数据 |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL \*NETDEV\_SOURCE\_DATA\_CALLBACK\_PF)  (  LPVOID lpPlayHandle,  const BYTE \*pucBuffer,  INT32 dwBufSize,  INT32 dwMediaDataType,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 当前实时预览句柄 |
| pucBuffer | IN | 存放拼帧前媒体流数据缓冲区 |
| dwBufSize | IN | 缓冲区大小 |
| dwMediaDataType | IN | 媒体数据类型，详见枚举 [NETDEV\_MEDIA\_DATA\_FORMAT\_E](#_媒体数据流格式) |
| lpUserParam | IN | 用户参数，即用户调用NETDEV\_RealPlay接口传入的lpUserData |

**Return Values：**

返回的启流句柄,返回0 表示失败,其他值表示返回的启流句柄值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* cbPlayDataCallBack回调函数中不能执行可能会占用时间较长的接口或操作；
* 该接口中的回调函数可以为空，这样该函数将不回调码流数据给用户，不过用户可以通过接口[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)注册捕获码流数据的回调；

**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)

### 停止实时预览

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopRealPlay  (  LPVOID lpPlayHandle  ); |

**接口描述：**

停止实时预览

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 预览句柄，[NETDEV\_RealPlay](#_实时预览)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_FastRealPlayByUrl](#_URL起流)

### 启动本地录像

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SaveRealData  (  LPVOID lpPlayHandle,  CHAR \*pszSaveFileName,  INT32 dwFormat  ); |

**接口描述：**

启动本地录像

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |
| pszSaveFileName | IN | 保存的文件名 |
| dwFormat | IN | 文件格式，详见枚举 [NETDEV\_MEDIA\_FILE\_FORMAT\_E](#_媒体文件格式枚举) |

**Remarks**：

* 起流句柄既可以为实况起流句柄，也可以为回放起流句柄；

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_StopSaveRealData](#_停止本地录像)

### 停止本地录像

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopSaveRealData  (  LPVOID lpPlayHandle  ); |

**接口描述：**

停止本地录像

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |

**Remarks**：

* 起流句柄既可以为实况起流句柄，也可以为回放起流句柄；

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SaveRealData](#_本地录像)

### 获取实况起流URL

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetStreamUrl  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwStreamType,  CHAR \*pszStreamUrl  ); |

**接口描述：**

启动实时预览

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwChannelID | IN | 通道号 |
| dwStreamType | IN | 码流类型，详见枚举 [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举) |
| pszStreamUrl | OUT | 码流URL，需用户分配内存，长度要求大于等于260 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* pszStreamUrl参数，需用户分配内存，长度要求大于等于260；

**See Also：**

[NETDEV\_FastRealPlayByUrl](#_URL起流_1)

### 通过URL启动实况预览

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

按照URL启动实时预览

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pszUrl | IN | 码流URL，通过[NETDEV\_GetStreamUrl](#_获取起流URL)获取 |
| pstPreviewInfo | IN | 预览参数，参考结构体 [NETDEV\_PREVIEWINFO\_S](#_实况预览参数结构体) |
| cbPlayDataCallBack | IN | 码流数据回调函数 |
| lpUserData | IN | 用户数据 |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL \*NETDEV\_SOURCE\_DATA\_CALLBACK\_PF)  (  LPVOID lpPlayHandle,  const BYTE \*pucBuffer,  INT32 dwBufSize,  INT32 dwMediaDataType,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 当前实时预览句柄 |
| pucBuffer | IN | 存放拼帧前媒体流数据缓冲区 |
| dwBufSize | IN | 缓冲区大小 |
| dwMediaDataType | IN | 媒体数据类型，详见枚举 [NETDEV\_MEDIA\_DATA\_FORMAT\_E](#_媒体数据流格式) |
| lpUserParam | IN | 用户参数，即用户调用NETDEV\_RealPlay接口传入的lpUserData |

**Return Values：**

返回的启流句柄,返回0 表示失败,其他值表示返回的启流句柄值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* cbPlayDataCallBack回调函数中不能执行可能会占用时间较长的接口或操作；
* 该接口中的回调函数可以为空，这样该函数将不回调码流数据给用户，不过用户可以通过接口[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)注册捕获码流数据的回调；

**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)

### 预览抓图

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_CapturePicture  (  LPVOID lpPlayHandle,  char \*pszFileName,  INT32 dwCaptureMode  ); |

**接口描述：**

按照URL启动实时预览

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |
| pszFileName | IN | 保存图像的文件路径，包括文件名 |
| dwCaptureMode | IN | 保存图像格式，详见枚举 [NETDEV\_PICTURE\_FORMAT\_E](#_抓图图片格式枚举) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 起流句柄既可以为实况起流句柄，也可以为回放起流句柄；
* 保存图像的文件路径中必须携带文件名；
* 文件名中可以不携带抓拍格式的后缀名；
* 抓拍文件保存路径，必须为已存在路径；

**See Also：**

[NETDEV\_CaptureNoPreview](#_非预览抓图)、[NETDEV\_CaptureNoPreviewV2](#_非预览获取图片数据)

### 非预览抓图

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_CaptureNoPreview  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwStreamType,  CHAR \*pszFileName,  INT32 dwCaptureMode  ); |

**接口描述：**

非预览下抓拍

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwChannelID | IN | 通道号 |
| dwStreamType | IN | 码流类型，详见枚举 [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举) |
| pszFileName | IN | 保存图像的文件路径，包括文件名 |
| dwCaptureMode | IN | 保存图像格式，详见枚举 [NETDEV\_PICTURE\_FORMAT\_E](#_抓图图片格式枚举) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 保存图像的文件路径中必须携带文件名；
* 文件名中可以不携带抓拍格式的后缀名；
* 当前仅支持JPG格式；
* 抓拍文件保存路径，必须为已存在路径；

**See Also：**

[NETDEV\_CapturePreview](#_预览抓图)、[NETDEV\_CaptureNoPreviewV2](#_非预览获取图片数据)

### 非预览获取图片数据

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_CaptureNoPreviewV2  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwStreamType,  INT32 dwCaptureMode,  INT32 dwPicBufferSize,  CHAR \*pszPicBuffer,  INT32 \*pdwPicRealSize  ); |

**接口描述：**

按照URL启动实时预览

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwChannelID | IN | 通道号 |
| dwStreamType | IN | 码流类型，详见枚举 [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举) |
| dwCaptureMode | IN | 保存图像格式，详见枚举 [NETDEV\_PICTURE\_FORMAT\_E](#_抓图图片格式枚举) |
| dwPicBufferSize | IN | 接收图片数据的Buffer大小 |
| pszPicBuffer | OUT | 接收图片数据的Buffer |
| pdwPicRealSize | OUT | 图片数据的实际大小 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 接收图片数据的buffer需用户自行分配内存，并将buffer大小赋值给dwPicBufferSize；
* 当前仅支持JPG格式；

**See Also：**

[NETDEV\_CapturePreview](#_预览抓图)、[NETDEV\_CaptureNoPreview](#_非预览抓图)

### 设置录像文件大小

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetRecordSize  (  LPVOID lpPlayHandle,  UINT32 ulFileSize  ); |

**接口描述：**

设置录像文件大小

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 用户播放句柄 |
| ulFileSize | IN | 录像文件大小（单位MB，范围从10MB到1024MB） |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

## 远程回放

### 录像查询

#### 查找录像文件列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindFile  (  LPVOID lpUserID,  [LPNETDEV\_FILECOND\_S](#_录像查找结构体) pstFindCond  ); |

**接口描述：**

根据文件类型.时间查找设备录像文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstFindCond | IN | 录像查询条件 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextFile](#_逐个获取查找到的文件信息)、[NETDEV\_FindClose](#_关闭录像文件查找)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextFile](#_逐个获取查找到的文件信息)、[NETDEV\_FindClose](#_关闭录像文件查找)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextFile](#_逐个获取查找到的文件信息)接口获取录像信息；
* 获取录像文件信息结束后必须调用[NETDEV\_FindClose](#_关闭录像文件查找)接口，以释放资源,关闭查找；

**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)

#### 逐个获取查找到的文件信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextFile  (  LPVOID lpFindHandle,  [LPNETDEV\_FINDDATA\_S](#_录像文件信息结构体) pstFindData  ); |

**接口描述：**

逐个获取查找到的录像文件信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 文件查找句柄，[NETDEV\_FindFile](#_查找录像文件列表)接口的返回值 |
| pstFindData | OUT | 录像文件信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindFile](#_查找录像文件列表)、[NETDEV\_FindClose](#_关闭录像文件查找)接口配套使用
* 此接口需多次调用，逐个获取录像文件信息；
* 获取录像文件信息结束后必须调用[NETDEV\_FindClose](#_关闭录像文件查找)接口，以释放资源,关闭查找；

**See Also：**

[NETDEV\_FindFile](#_查找录像文件列表)、[NETDEV\_FindClose](#_关闭录像文件查找)

#### 关闭录像文件查找

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindClose  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭录像文件查找,释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 文件查找句柄，[NETDEV\_FindFile](#_查找录像文件列表)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindFile](#_查找录像文件列表)、[NETDEV\_FindNextFile](#_逐个获取查找到的文件信息)接口配套使用
* 获取录像文件信息结束后必须调用[NETDEV\_FindClose](#_关闭录像文件查找)接口，以释放资源,关闭查找；

**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)

### 获取录像天数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetVideoDayNums  (  LPVOID lpUserID,  INT32 dwChannelID,  UINT32 \*dwDayNums  ); |

**接口描述：**

查询指定通道录像的天数

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwChannelID | IN | 通道号 |
| dwDayNums | OUT | 录像天数 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_QuickSearch](#_单通道查询)、[NETDEV\_QuickSearchMultiChl](#_多通道查询)

### 查询指定月份的录像分布信息

#### 单通道查询

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

查询指定月份的录像分布信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwChannelID | IN | 通道号 |
| pstMonthInfo | IN | 月份信息 |
| pstMonthStatus | OUT | 录像分布状态 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 此接口只能查询单个通道的录像分布信息

**See Also：**

[NETDEV\_GetVideoDayNums](#_获取录像天数)、[NETDEV\_QuickSearchMultiChl](#_多通道查询)

#### 多通道查询

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

查询多个通道指定月份的录像分布信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstChannels | IN | 通道列表 |
| pstMonthInfo | IN | 月份信息 |
| pstMonthStatus | OUT | 录像分布状态 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_QuickSearch](#_单通道查询)、[NETDEV\_GetVideoDayNums](#_获取录像天数)

### 获取回放URL

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetReplayUrl\_V30  (  LPVOID lpUserID,  [LPNETDEV\_PLAYBACKCOND\_S](#_录像回放参数结构体) pstPlayBackCond,  CHAR\* pszUrl  ); |

**接口描述：**

获取回放URL

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstPlayBackCond | IN | 录像回放参数 |
| pszUrl | OUT | 码流URL，内存长度要求大于等于260 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_FastPlayBackByUrl](#_根据URL回放录像文件)

### 根据URL回放录像文件

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FastPlayBackByUrl  (  LPVOID lpUserID,  CHAR \*pszUrl,  [LPNETDEV\_PLAYBACKCOND\_S](#_录像回放参数结构体) pstPlayBackParam  ); |

**接口描述：**

根据URL回放录像文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pszUrl | IN | 码流URL |
| pstPlayBackParam | IN | 录像回放参数结构体 |

**Return Values：**

返回的启流句柄,返回0 表示失败,其他值表示返回的启流句柄值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetReplayUrl](#_获取回放URL)\_V30

### 按文件名回放录像文件

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_PlayBackByName  (  LPVOID lpUserID,  [LPNETDEV\_PLAYBACKINFO\_S](#_按文件名回放录像参数结构体) pstPlayBackInfo  ); |

**接口描述：**

按名称回放录像文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstPlayBackInfo | IN | 录像回放信息 |

**Return Values：**

返回的启流句柄,返回0 表示失败,其他值表示返回的启流句柄值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PlayBackByTime](#_按时间回放录像文件)

### 按时间回放录像文件

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_PlayBackByTime  (  LPVOID lpUserID,  [LPNETDEV\_PLAYBACKCOND\_S](#_录像回放参数结构体) pstPlayBackInfo  ); |

**接口描述：**

按时间回放录像文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstPlayBackInfo | IN | 录像回放信息 |

**Return Values：**

返回的启流句柄,返回0 表示失败,其他值表示返回的启流句柄值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PlayBackByName](#_按文件名回放录像文件)

### 单帧回放

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetOneFramePlayWithTime  (  LPVOID lpPlayHandle,  INT64 dwPlayTime  ); |

**接口描述：**

设置特定时间点单帧回放

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 回放句柄 |
| dwPlayTime | IN | 特定时间点 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 回放控制

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PlayBackControl  (  LPVOID lpPlayHandle,  INT32 dwControlCode,  LPVOID lpBuffer  ); |

**接口描述：**

控制录像回放的状态

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 回放句柄 |
| dwControlCode | IN | 回放控制命令，详见枚举 [NETDEV\_VOD\_PLAY\_CTRL\_E](#_回放控制命令枚举) |
| lpBuffer | INOUT | 回放控制输入/输出参数，若为播放速度，则详见枚举 [NETDEV\_VOD\_PLAY\_STATUS\_E](#_播放/下载速度枚举) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks：**

* 开始、暂停、恢复播放时，lpBuffer置为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)

### 停止回放

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopPlayBack  (  LPVOID lpPlayHandle  ); |

**接口描述：**

停止回放业务

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 回放句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 按文件名下载录像文件

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

按文件名下载录像文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstPlayBackInfo | IN | 录像回放参数 |
| pszSaveFileName | IN | 保存的文件路径（绝对路径，包含文件名） |
| dwFormat | IN | 录像文件保存格式，详见枚举[NETDEV\_MEDIA\_FILE\_FORMAT\_E](#_媒体文件格式枚举) |

**Return Values：**

返回的启流句柄,返回0 表示失败,其他值表示返回的启流句柄值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 保存的文件路径必须为绝对路径，且携带文件名；
* 抓拍文件保存路径，必须为已存在路径；

**See Also：**

[NETDEV\_GetFileByTime](#_按时间下载录像文件)、[NETDEV\_StopGetFile](#_停止下载录像文件)

### 按时间下载录像文件

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_GetFileByTime  (  LPVOID lpUserID,  [LPNETDEV\_PLAYBACKCOND\_S](#_录像回放参数结构体) pstPlayBackCond,  CHAR \*pszSaveFileName,  INT32 dwFormat  ); |

**接口描述：**

按时间下载录像文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstPlayBackCond | IN | 录像回放参数 |
| pszSaveFileName | IN | 保存的文件路径（绝对路径，包含文件名） |
| dwFormat | IN | 录像文件保存格式，详见枚举[NETDEV\_MEDIA\_FILE\_FORMAT\_E](#_媒体文件格式枚举) |

**Return Values：**

返回的启流句柄,返回0 表示失败,其他值表示返回的启流句柄值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 保存的文件路径必须为绝对路径，且携带文件名；
* 抓拍文件保存路径，必须为已存在路径；

**See Also：**

[NETDEV\_GetFileByName](#_按文件名下载录像文件)、[NETDEV\_StopGetFile](#_停止下载录像文件)

### 停止下载录像文件

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopGetFile  (  LPVOID lpPlayHandle  ); |

**接口描述：**

停止下载录像文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 录像下载函数句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 查询录像起止时间

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

查询录像起止时间

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstChlList | IN | 通道列表 |
| pstRecordTimeList | OUT | 录像时间列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

## 码流数据回调

### 注册原始码流回调

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPlayDataCallBack  (  LPVOID lpPlayHandle,  NETDEV\_SOURCE\_DATA\_CALLBACK\_PF cbPlayDataCallBack,  INT32 bContinue,  LPVOID lpUser  ); |

**接口描述：**

注册码流回调函数:原始码流；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| cbPlayDataCallBack | IN | 码流数据回调函数 |
| bContinue | IN | 是否继续进行后面的拼帧.解码和显示操作 |
| lpUser | IN | 用户数据 |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL \*NETDEV\_SOURCE\_DATA\_CALLBACK\_PF)  (  LPVOID lpPlayHandle,  const BYTE \*pucBuffer,  INT32 dwBufSize,  INT32 dwMediaDataType,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 当前实时预览句柄/回放句柄 |
| pucBuffer | IN | 存放拼帧前媒体流数据缓冲区 |
| dwBufSize | IN | 缓冲区大小 |
| dwMediaDataType | IN | 媒体数据类型，详见枚举 [NETDEV\_MEDIA\_DATA\_FORMAT\_E](#_媒体数据流格式) |
| lpUserParam | IN | 用户数据，在调用NETDEV\_SetPlayDataCallBack函数时指定的用户参数 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 若关闭回调函数,将第二个参数置为NULL；
* 用户应及时处理输出的数据,确保函数尽快返回,否则会影响播放器内的媒体流处理；

**See Also：**

[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_FastRealPlayByUrl](#_通过URL启动实况预览)

### 注册解码后音频数据回调

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPlayDecodeAudioCB  (  LPVOID lpPlayHandle,  NETDEV\_DECODE\_AUDIO\_DATA\_CALLBACK\_PF cbPlayDecodeAudioCallBack,  INT32 bContinue,  LPVOID lpUserData  ); |

**接口描述：**

注册码流回调函数：解码后音频媒体流数据；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| cbPlayDecodeAudioCallBack | IN | 码流数据回调函数 |
| bContinue | IN | 是否继续进行播放操作 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 当前实时预览句柄/回放句柄 |
| pstWaveData | IN | 存放解码后音频流数据信息缓冲区，详见结构体[NETDEV\_WAVE\_DATA\_S](#_音频数据结构体) |
| lpUserParam | IN | 用户数据，在调用NETDEV\_SetPlayDecodeAudioCB时指定的用户参数 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 若关闭回调函数,将第二个参数置为NULL；
* 用户应及时处理输出的数据,确保函数尽快返回,否则会影响播放器内的媒体流处理；

**See Also：**

[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_FastRealPlayByUrl](#_通过URL启动实况预览)

### 注册解码后视频数据回调

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPlayDecodeVideoCB  (  LPVOID lpPlayHandle,  NETDEV\_DECODE\_VIDEO\_DATA\_CALLBACK\_PF cbPlayDecodeVideoCALLBACK,  INT32 bContinue,  LPVOID lpUserData  ); |

**接口描述：**

注册码流回调函数：解码后视频媒体流数据；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| cbPlayDecodeVideoCALLBACK | IN | 码流数据回调函数 |
| bContinue | IN | 是否继续进行后面的显示操作 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 当前实时预览句柄/回放句柄 |
| pstPictureData | IN | 存放解码后视频流数据信息缓冲区，详见结构体[NETDEV\_PICTURE\_DATA\_S](#_图像数据结构体) |
| lpUserParam | IN | 用户数据，在调用NETDEV\_SetPlayDecodeVideoCB时指定的用户参数 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 若关闭回调函数,将第二个参数置为NULL；
* 用户应及时处理输出的数据,确保函数尽快返回,否则会影响播放器内的媒体流处理；

**See Also：**

[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_FastRealPlayByUrl](#_通过URL启动实况预览)

### 注册拼帧后音频数据回调

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPlayParseAudioCB  (  LPVOID lpPlayHandle,  NETDEV\_PARSE\_AUDIO\_DATA\_CALLBACK\_PF cbPlayParseAudioCallBack,  INT32 bContinue,  LPVOID lpUserData  ); |

**接口描述：**

注册码流回调函数：拼帧后音频媒体流数据；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| cbPlayParseAudioCallBack | IN | 码流数据回调函数 |
| bContinue | IN | 是否继续进行播放操作 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 当前实时预览句柄/回放句柄 |
| pstParseAudioData | IN | 存放拼帧后音频流数据信息缓冲区，详见结构体 [NETDEV\_PARSE\_AUDIO\_DATA\_S](#_拼帧后音频数据结构体) |
| lpUserParam | IN | 用户数据，在调用NETDEV\_SetPlayParseAudioCB时指定的用户参数 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 若关闭回调函数,将第二个参数置为NULL；
* 用户应及时处理输出的数据,确保函数尽快返回,否则会影响播放器内的媒体流处理；

**See Also：**

[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_FastRealPlayByUrl](#_通过URL启动实况预览)

### 注册拼帧后视频数据回调

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPlayParseCB  (  LPVOID lpPlayHandle,  NETDEV\_PARSE\_VIDEO\_DATA\_CALLBACK\_PF cbParsePlayDataCallBack,  INT32 bContinue,  LPVOID lpUserData  ); |

**接口描述：**

注册码流回调函数：拼帧后视频媒体流数据；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| cbParsePlayDataCallBack | IN | 码流数据回调函数 |
| bContinue | IN | 是否继续进行后面的解码和显示操作 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 当前实时预览句柄/回放句柄 |
| pstParseVideoData | IN | 存放拼帧后视频流数据信息缓冲区，详见结构体 [NETDEV\_PARSE\_VIDEO\_DATA\_S](#_视频解析数据结构体) |
| lpUserParam | IN | 用户数据，在调用NETDEV\_SetPlayParseCB时指定的用户参数 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 若关闭回调函数,将第二个参数置为NULL；
* 用户应及时处理输出的数据,确保函数尽快返回,否则会影响播放器内的媒体流处理；

**See Also：**

[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_FastRealPlayByUrl](#_通过URL启动实况预览)

### 注册画图回调

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPlayDisplayCB  (  LPVOID lpPlayHandle,  NETDEV\_DISPLAY\_CALLBACK\_PF cbPlayDisplayCallBack,  LPVOID lpUserData  ); |

**接口描述：**

设置画图回调，即显示后数据回调；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| cbPlayDisplayCallBack | IN | 码流数据回调函数 |
| lpUserData | IN | 用户数据 |

**Callback Function**

|  |
| --- |
| typedef void (STDCALL\* NETDEV\_DISPLAY\_CALLBACK\_PF)  (  LPVOID lpHandle,  LPVOID hdc,  LPVOID lpUserParam  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpHandle | IN | 实时预览句柄/回放句柄 |
| hdc | IN | 显示表面的hdc,用户可以用这个hdc 作图 |
| lpUserParam | IN | 用户设置参数，即调用NETDEV\_ SetPlayDisplayCB函数时指定的用户参数 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 若关闭回调函数,将第二个参数置为NULL；
* 用户应及时处理输出的数据,确保函数尽快返回,否则会影响播放器内的媒体流处理；

**See Also：**

[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_FastRealPlayByUrl](#_通过URL启动实况预览)

## 视频参数配置

### 获取窗口丢包率

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetLostPacketRate  (  LPVOID lpPlayHandle,  INT32 \*pulRecvPktNum,  INT32 \*pulLostPktNum  ); |

**接口描述：**

获取窗口丢包率

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| pulRecvPktNum | OUT | 接收的数据包数量 |
| pulLostPktNum | OUT | 丢失的数据包数量 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 具体丢包率需要用户根据丢失的数据包数量/接收的数据包数量自行计算；

**See Also：**

[NETDEV\_ResetLostPacketRate](#_重置窗口丢包率)

### 重置窗口丢包率

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ResetLostPacketRate  (  LPVOID lpPlayHandle  ); |

**接口描述：**

重置窗口丢包率

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetLostPacketRate](#_获取窗口丢包率)

### 获取窗口码率

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetBitRate  (  LPVOID lpPlayHandle,  INT32 \*pdwBitRate  ); |

**接口描述：**

获取窗口码率

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| pdwBitRate | OUT | 当前播放窗口码率 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取窗口帧率

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetFrameRate  (  LPVOID lpPlayHandle,  INT32 \*pdwFrameRate  ); |

**接口描述：**

获取窗口帧率

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| pdwFrameRate | OUT | 当前播放窗口帧率 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取窗口分辨率

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetResolution  (  LPVOID lpPlayHandle,  INT32 \*pdwWidth,  INT32 \*pdwHeight  ); |

**接口描述：**

获取当前播放窗口视频分辨率

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| pdwWidth | OUT | 获取的分辨率-宽度 |
| pdwHeight | OUT | 获取的分辨率-高度 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取影像参数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetVideoEffect  (  LPVOID lpPlayHandle,  [LPNETDEV\_VIDEO\_EFFECT\_S](#_影像信息结构体) pstImageInfo  ); |

**接口描述：**

获取当前画面影像参数

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| pstImageInfo | OUT | 画面影像参数，详见结构体 [NETDEV\_VIDEO\_EFFECT\_S](#_影像信息结构体) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 只获取当前画面影像参数；

### 影像调节

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetVideoEffect  (  LPVOID lpPlayHandle,  [LPNETDEV\_VIDEO\_EFFECT\_S](#_影像信息结构体) pstImageInfo  ); |

**接口描述：**

当前画面影像参数调节

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| pstImageInfo | IN | 画面影像参数，详见结构体 [NETDEV\_VIDEO\_EFFECT\_S](#_影像信息结构体) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 影像调节，只改变当前画面；

### 获取窗口编码格式

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetVideoEncodeFmt  (  LPVOID lpPlayHandle,  INT32 \*pdwVideoEncFmt  ); |

**接口描述：**

获取当前播放窗口编码格式

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| pdwVideoEncFmt | OUT | 视频编码格式，详见枚举 [NETDEV\_VIDEO\_CODE\_TYPE\_E](#_视频编码格式枚举) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 动态产生一个关键帧

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_MakeKeyFrame  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwStreamType  ); |

**接口描述：**

动态产生一个关键帧，即：强制I帧

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwChannelID | IN | 通道号 |
| dwStreamType | IN | 码流类型，详见枚举 [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 设置数字放大

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetDigitalZoom  (  LPVOID lpPlayHandle,  LPVOID hWnd,  [LPNETDEV\_RECT\_S](#_矩形区域结构体) pstRect  ); |

**接口描述：**

设置数字放大

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| hWnd | IN | 窗口句柄 |
| pstRect | IN | 数字放大的矩形区域 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* pstRect为空时,显示全部画面,即退出数字放大；

### 设置元数据处理开关

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetIVAEnable  (  LPVOID lpPlayHandle,  BOOL bEnableIVA  ); |

**接口描述：**

设置元数据处理开关

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄 |
| bEnableIVA | IN | 是否添加元数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also**：

[NETDEV\_SetIVAShowParam](#_设置元数据处理参数)

### 设置元数据处理参数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetIVAShowParam  (  INT32 dwShowParam  ); |

**接口描述：**

设置元数据绘图显示类型

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| dwShowParam | IN | 元数据显示类型，详见枚举 [NETDEV\_IVA\_SHOW\_RULE\_E](#_元数据显示类型) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also**：

[NETDEV\_SetIVAEnable](#_设置元数据处理开关)

### 设置视频流畅性

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPictureFluency  (  LPVOID lpPlayHandle,  INT32 dwFluency  ); |

**接口描述：**

设置元数据绘图显示类型

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| dwFluency | IN | 图像播放流畅性优先类型，详见枚举 [NETDEV\_PICTURE\_FLUENCY\_E](#_图像播放流畅性枚举) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 设置视频图像显示比例

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetRenderScale  (  LPVOID lpPlayHandle,  [NETDEV\_RENDER\_SCALE\_E](#_视频显示比例枚举) enRenderScalesss  ); |

**接口描述：**

设置视频图像显示比例

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄/回放句柄 |
| enRenderScale | IN | 视频图像的显示比例 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

## 语音功能

### 开启语音对讲

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_StartVoiceCom  (  LPVOID lpUserID,  INT32 dwChannelID,  NETDEV\_DECODE\_AUDIO\_DATA\_CALLBACK\_PF cbPlayDataCallBack,  LPVOID lpUserData  ); |

**接口描述：**

开启语音对讲功能

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwChannelID | IN | 通道号 |
| cbPlayDataCallBack | IN | 解码后音频流数据回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 当前语音对讲接口返回句柄 |
| pstWaveData | IN | 存放解码后音频流数据信息，详见结构体[NETDEV\_WAVE\_DATA\_S](#_音频数据结构体) |
| lpUserParam | IN | 用户参数，即用户调用NETDEV\_StartVoiceCom接口传入的lpUserData |

**Return Values：**

返回0表示失败,其他值作为[NETDEV\_StopVoiceCom](#_停止语音对讲)等函数句柄参数。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* cbPlayDataCallBack函数实现不可占用过长时间；
* 该接口中的回调函数和用户参数可以为空；
* 通道号设备本身为0，设备下通道以SDK接口查询到的通道号为准；

**See Also：**

[NETDEV\_StopVoiceCom](#_停止语音对讲)

### 停止语音对讲

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopVoiceCom (LPVOID lpPlayHandle); |

**接口描述：**

停止语音对讲功能

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 语音对讲句柄，[NETDEV\_StartVoiceCom](#_开启语音对讲)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_StartVoiceCom](#_开启语音对讲)

### 开启语音数据转发

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_StartInputVoiceSrv  (  LPVOID lpUserID,  INT32 dwChannelID  ); |

**接口描述：**

开启语音数据转发功能

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwChannelID | IN | 通道号 |

**Return Values：**

返回0表示失败,其他值作为[NETDEV\_StopInputVoiceSrv](#_设置解析后语音数据)等函数句柄参数。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 通道号设备本身为0，设备下通道以SDK接口查询到的通道号为准；

**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)

### 语音数据转发

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_InputVoiceData  (  LPVOID lpVoiceComHandle,  LPVOID lpDataBuf,  INT32 dwDataLen,  [LPNETDEV\_AUDIO\_SAMPLE\_PARAM\_S](#_音频参数结构体) pstVoiceParam  ); |

**接口描述：**

开启语音数据转发功能

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpVoiceComHandle | IN | 用户登录句柄，[NETDEV\_StartInputVoiceSrv](#_开启语音数据转发)接口的返回值 |
| lpDataBuf | IN | PCM格式音频数据 |
| dwDataLen | IN | 音频数据长度 |
| pstVoiceParam | IN | 音频参数 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 停止语音数据转发

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopInputVoiceSrv (LPVOID lpVoiceComHandle); |

**接口描述：**

停止语音对讲功能

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpVoiceComHandle | IN | 开启语音数据转发句柄，[NETDEV\_StartInputVoiceSrv](#_开启语音数据转发)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_StopInputVoiceSrv](#_停止语音数据转发)

### 开启语音广播

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_StartVoiceBroadcast  (  LPVOID lpUserID,  INT32 dwChannelID  ); |

**接口描述：**

开启语音广播功能

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwChannelID | IN | 通道号 |

**Return Values：**

返回0表示失败,其他值作为[NETDEV\_StopVoiceBroadcast](#_停止语音广播)等函数句柄参数。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 通道号设备本身为0，设备下通道以SDK接口查询到的通道号为准；

**See Also：**

[NETDEV\_StopVoiceBroadcast](#_停止语音广播)

### 停止语音广播

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopVoiceBroadcast (LPVOID lpPlayHandle); |

**接口描述：**

停止语音广播功能

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 语音广播句柄，  [NETDEV\_StartVoiceBroadcast](#_开启语音广播)、[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)  接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_StartVoiceBroadcast](#_开启语音广播)、[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)

### 开启语音广播组

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_CreateVoiceBroadcastGroup  (  LPVOID lpUserID,  [LPNETDEV\_OPERATE\_LIST\_S](#_批处理列表) pstChnList  )； |

**接口描述：**

开启语音广播组功能

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstChnList | IN | 通道列表 |

**Return Values：**

返回0表示失败,其他值作为[NETDEV\_StopVoiceBroadcast](#_停止语音广播)等函数句柄参数。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 仅VMS设备支持；

**See Also：**

[NETDEV\_ModifyVoiceBroadcastGroup](#_修改语音广播组)

### 修改语音广播组

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyVoiceBroadcastGroup  (  LPVOID lpPlayHandle,  [LPNETDEV\_MODIFY\_VOICE\_BROADCAST\_INFO\_S](#_语音广播组信息结构体) pstModifyInfo  ); |

**接口描述：**

修改语音广播组功能

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 语音广播组句柄，[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)接口的返回值 |
| pstModifyInfo | IN | 修改语音广播组信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 仅VMS设备支持；

**See Also：**

[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)

### 获取语音广播组通道状态

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取语音广播组通道的状态

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 语音广播组句柄，[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)接口的返回值 |
| pstGroupInfo | OUT | 语音广播组信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 仅VMS设备支持；

**See Also：**

[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)

### 修改语音广播组通道状态

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

修改语音广播组通道的状态

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 语音广播组句柄，[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)接口的返回值 |
| pstCtrlInfo | IN | 语音广播组控制信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 仅VMS设备支持；

**See Also：**

[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)

### 开启扬声器声音

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_OpenSound(IN LPVOID lpPlayHandle); |

**接口描述：**

开启扬声器声音

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |

**Remarks**：

* 起流句柄可以为[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_StartVoiceCom](#_开启语音对讲)等启流接口返回值

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 获取扬声器音量

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetSoundVolume  (  LPVOID lpPlayHandle,  INT32\* pdwVolume  ); |

**接口描述：**

获取扬声器音量

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |
| pdwVolume | OUT | 音量大小 |

**Remarks**：

* 起流句柄可以为[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_StartVoiceCom](#_开启语音对讲)等启流接口返回值

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 设置扬声器音量

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SoundVolumeControl  (  LPVOID lpPlayHandle,  INT32 dwVolume  ); |

**接口描述：**

设置扬声器音量

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |
| dwVolume | IN | 音量大小，取值范围[0-255] |

**Remarks**：

* 起流句柄可以为[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_StartVoiceCom](#_开启语音对讲)等启流接口返回值

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 关闭扬声器声音

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_CloseSound(LPVOID lpPlayHandle) |

**接口描述：**

关闭扬声器声音

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |

**Remarks**：

* 起流句柄可以为[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_StartVoiceCom](#_开启语音对讲)等启流接口返回值

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 设置静音状态

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetMuteStatus  (  LPVOID lpPlayHandle,  BOOL bMute  ); |

**接口描述：**

设置静音状态

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |
| bMute | IN | 静音状态，0关闭静音，1开启静音 |

**Remarks**：

* 起流句柄可以为[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_StartVoiceCom](#_开启语音对讲)等启流接口返回值

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetMuteStatus](#_获取静音状态)

### 获取静音状态

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetMuteStatus  (  LPVOID lpPlayHandle,  BOOL\* pbMute  ); |

**接口描述：**

获取静音状态

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |
| pbMute | OUT | 静音状态，0关闭静音，1开启静音 |

**Remarks**：

* 起流句柄可以为[NETDEV\_RealPlay](#_实时预览)、[NETDEV\_StartVoiceCom](#_开启语音对讲)等启流接口返回值

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SetMuteStatus](#_设置静音状态)

### 开启麦克风

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_OpenMic (LPVOID lpPlayHandle); |

**接口描述：**

开启麦克风

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |

**Remarks**：

* 起流句柄可以为[NETDEV\_StartVoiceCom](#_开启语音对讲)、[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)等启流接口返回值

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 获取麦克风音量

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetMicVolume  (  LPVOID lpPlayHandle，  INT32\* pdwVolume  ); |

**接口描述：**

获取麦克风音量

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |
| pdwVolume | OUT | 音量大小，取值范围[0-255] |

**Remarks**：

* 起流句柄可以为[NETDEV\_StartVoiceCom](#_开启语音对讲)、[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)等启流接口返回值

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 设置麦克风音量

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_MicVolumeControl  (  LPVOID lpPlayHandle，  INT32 dwVolume  ); |

**接口描述：**

设置麦克风音量

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |
| dwVolume | OUT | 音量大小，取值范围[0-255] |

**Remarks**：

* 起流句柄可以为[NETDEV\_StartVoiceCom](#_开启语音对讲)、[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)等启流接口返回值

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 关闭麦克风

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_CloseMic (IN LPVOID lpPlayHandle); |

**接口描述：**

关闭麦克风

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 起流句柄 |

**Remarks**：

* 起流句柄可以为[NETDEV\_StartVoiceCom](#_开启语音对讲)、[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)等启流接口返回值

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

## 本地播放

### 打开本地文件

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_OpenMediaFile(CHAR \*pszFilename); |

**接口描述：**

打开本地文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| pszFilename | IN | 文件名，包含绝对路径，建议长度不超过256字节 |

**Return Values：**

返回0表示失败,其他值作为[NETDEV\_StartPlayMediaFile](#_开始播放本地文件)等函数句柄参数。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 开始播放本地文件

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StartPlayMediaFile  (  LPVOID lpPlayHandle,  LPVOID lpPlayWnd  ); |

**接口描述：**

开始播放本地文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 文件句柄,调用[NETDEV\_OpenMediaFile](#_打开本地文件)的返回值 |
| lpPlayWnd | IN | 播放窗口句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 播放过程中，可以调用回放控制[NETDEV\_PlayBackControl](#_回放控制)

**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)

### 停止播放本地文件

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopPlayMediaFile(LPVOID lpPlayHandle); |

**接口描述：**

停止播放本地文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 文件句柄,调用[NETDEV\_OpenMediaFile](#_打开本地文件)的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 获取本地文件播放总时长

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetMediaFileTime  (  LPVOID lpPlayHandle,  INT32\* pdwTotalTime  ); |

**接口描述：**

停止播放本地文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 文件句柄,调用[NETDEV\_OpenMediaFile](#_打开本地文件)的返回值 |
| pdwTotalTime | OUT | 本地文件播放时间总长度（s） |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

## 鱼眼矫正

### 鱼眼码流判断

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_IsFishEyeStream  (  LPVOID lpPlayHandle,  BOOL \*pbFishEyeStream  ) |

**接口描述：**

判断是否为鱼眼码流

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 预览句柄，[NETDEV\_RealPlay](#_实时预览)接口的返回值 |
| pbFishEyeStream | OUT | 是否为鱼眼码流 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 设置5ePTZ显示模式

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPtzDisplayMode  (  LPVOID lpPlayHandle,  UINT32 udwPtzMode  ); |

**接口描述：**

设置5ePTZ显示模式

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 预览句柄，[NETDEV\_RealPlay](#_实时预览)接口的返回值 |
| udwPtzMode | IN | 5ePTZ显示模式 参考# [NETDEV\_PTZ\_DISPLAY\_MODE](#_5ePTZ显示模式枚举定义) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 鼠标操作模式

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetMouseMoveMode  (  LPVOID lpPlayHandle,  INT32 dwOperateMode,  UINT32 udwFlag,  INT16 wDelta,  [LPNETDEV\_POINT\_S](#_点坐标结构体) pstPoint  ); |

**接口描述：**

鼠标操作模式

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 预览句柄，[NETDEV\_RealPlay](#_实时预览)接口的返回值 |
| dwOperateMode | IN | 鼠标操作方式,参见#[NETDEV\_MOUSE\_MOVE\_MODE\_E](#_鼠标移动模式枚举) |
| udwFlag | IN | 与实际标志位,预留 |
| wDelta | IN | 鼠标滚轮偏移度(可选) |
| pstPoint | IN | 位移坐标 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取鱼眼矫正参数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetFishEyeParam  (  LPVOID lpPlayHandle,  [LPNETDEV\_POINT\_S](#_点坐标结构体) pstCenterPoint,  UINT32 \*pudwRadius,  LPVOID lpParm  ); |

**接口描述：**

获取鱼眼矫正参数

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 预览句柄，[NETDEV\_RealPlay](#_实时预览)接口的返回值 |
| pstCenterPoint | OUT | 鱼眼图像圆的圆心 |
| pudwRadius | OUT | 鱼眼图像圆的半径 |
| lpParm | OUT | 预留鱼眼镜头参数 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 设置模式和安装位置

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPtzAndFixMode  (  LPVOID lpPlayHandle,  INT32 dwPtzMode,  INT32 dwInstallMode  ) |

**接口描述：**

设置模式和安装位置

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 预览句柄，[NETDEV\_RealPlay](#_实时预览)接口的返回值 |
| dwPtzMode | IN | 鱼眼PTZ模式,参考# [NETDEV\_FISHEYE\_PTZ\_MODE\_E](#_鱼眼设备矫正模式枚举) |
| dwInstallMode | IN | 安装位置,参考# [NETDEV\_INSTALL\_MODE\_E](#_设备安装模式枚举定义) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetPtzAndFixMode](#_获取模式和安装位置)

### 获取模式和安装位置

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetPtzAndFixMode  (  LPVOID lpPlayHandle,  INT32 dwPtzMode,  INT32 dwInstallMode  ) |

**接口描述：**

鼠标操作模式

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 预览句柄，[NETDEV\_RealPlay](#_实时预览)接口的返回值 |
| dwPtzMode | OUT | 鱼眼PTZ模式,参考# [NETDEV\_FISHEYE\_PTZ\_MODE\_E](#_鱼眼设备矫正模式枚举) |
| dwInstallMode | OUT | 安装位置,参考# [NETDEV\_INSTALL\_MODE\_E](#_设备安装模式枚举) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SetPtzAndFixMode](#_设置模式和安装位置)

## 手动录像

### 获取手动录像状态

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetManualRecordStatus  (  LPVOID lpUserID,  [LPNETDEV\_MANUAL\_RECORD\_CFG\_S](#_手动录像结构体) pstManualRecordCfg,  UINT32 \*pudwRecodeStatus  ); |

**接口描述：**

获取手动录像状态

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstManualRecordCfg | IN | 手动录像参数信息 |
| pudwRecodeStatus | OUT | 手动录像状，参见枚举[NETDEV\_MANUAL\_RECORD\_STATUS\_E](#_录像状态枚举_1) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 开启手动录像

|  |
| --- |
| BOOL STDCALL NETDEV\_StartManualRecord  (  LPVOID lpUserID,  [LPNETDEV\_MANUAL\_RECORD\_CFG\_S](#_手动录像结构体) pstManualRecordCfg  ); |

**接口描述：**

开启手动录像

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstManualRecordCfg | IN | 手动录像参数信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See also：**

[NETDEV\_StopManualRecord](#_停止手动录像)

### 停止手动录像

|  |
| --- |
| BOOL STDCALL NETDEV\_StopManualRecord  (  LPVOID lpUserID,  [LPNETDEV\_MANUAL\_RECORD\_CFG\_S](#_手动录像结构体) pstManualRecordCfg  ); |

**接口描述：**

停止手动录像

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstManualRecordCfg | IN | 手动录像参数信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See also：**

[NETDEV\_StartManualRecord](#_开启手动录像)

## 云台控制

### 云台3D定位

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZ3DPosition  (  LPVOID lpPlayHandle,  [LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S](#_拉框缩放结构体) pstZoomAreaInfo  ); |

**接口描述：**

云台3D定位(需要启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄 |
| pstZoomAreaInfo | IN | 拉框放大结构体 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZ3DPosition\_Other](#_云台3D定位)

### 无预览云台3D定位

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZ3DPosition\_Other  (  LPVOID                                                         lpUserID,  INT32                                                            dwChannelID,  [LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S](#_拉框缩放结构体)    pstZoomAreaInfo  ); |

**接口描述：**

云台3D定位(不需要启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChannelID | IN | 通道号 |
| pstZoomAreaInfo | IN | 拉框放大结构体 |

**Remarks**：

* 无需启动实况预览

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZ3DPosition](#_云台3D定位_1)

### 云台拉框放大

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZAreaZoomIn  (  LPVOID lpPlayHandle,  [LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S](#_拉框缩放结构体) pstZoomAreaInfo  )； |

**接口描述：**

云台拉框放大(需要启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄 |
| pstZoomAreaInfo | IN | 拉框缩放结构体 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZAreaZoomIn\_Other](#_云台拉框放大)

### 无预览云台拉框放大

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZAreaZoomIn\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S](#_拉框缩放结构体) pstZoomAreaInfo  ); |

**接口描述：**

云台拉框放大(不需要启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChannelID | IN | 通道ID |
| pstZoomAreaInfo | IN | 拉框缩放结构体 |

**Remarks**：

* 无需启动实况预览

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZAreaZoomIn](#_云台拉框放大_1)

### 云台拉框缩小

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZAreaZoomOut  (  LPVOID lpPlayHandle,  [LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S](#_拉框缩放结构体) pstZoomAreaInfo  ); |

**接口描述：**

云台拉框缩小(需要启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄 |
| pstZoomAreaInfo | IN | 拉框缩放结构体 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZAreaZoomOut\_Other](#_拉框放大.缩小)

### 无预览云台拉框缩小

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZAreaZoomOut\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_ZOOM\_AREA\_INFO\_S](#_拉框缩放结构体) pstZoomAreaInfo  ); |

**接口描述：**

云台拉框缩小（不需要启动预览）

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChannelID | IN | 通道ID |
| pstZoomAreaInfo | IN | 拉框缩放结构体 |

**Remarks**：

* 无需启动实况预览

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZAreaZoomOut](#_云台拉框缩小)

### 拉框放大.缩小

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZSelZoomIn  (  LPVOID lpPlayHandle,  [LPNETDEV\_PTZ\_OPERATEAREA\_S](#_拉框放大结构体_1) pstPtzOperateArea  ); |

**接口描述：**

拉框放大.缩小(需先启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄 |
| pstPtzOperateArea | IN | 拉框放大结构体信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZSelZoomIn\_Other](#_拉框放大.缩小)

### 无预览拉框放大.缩小

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZSelZoomIn\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_OPERATEAREA\_S](#_拉框放大结构体_1) pstPtzOperateArea  ); |

**接口描述：**

拉框放大.缩小(不需先启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| pstPtzOperateArea | IN | 图像区域信息。注意参数取值范围 |

**Remarks**：

* 无需启动实况预览

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZSelZoomIn](#_拉框放大.缩小_1)

### 云台控制操作

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZControl  (  LPVOID lpPlayHandle,  INT32 dwPTZCommand,  INT32 dwSpeed  )； |

**接口描述：**

云台控制操作(需先启动实况预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄 |
| dwPTZCommand | IN | 云台控制命令，参考枚举[NETDEV\_PTZ\_E](#_云台命令枚举) |
| dwSpeed | IN | 云台控制的速度，取值范围[1,9] |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZControl\_Other](#_云台控制操作)

### 无预览云台控制操作

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZControl\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwPTZCommand,  INT32 dwSpeed  ); |

**接口描述：**

云台控制操作(无需先启动实况预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| dwPTZCommand | IN | 云台控制命令，参考枚举[NETDEV\_PTZ\_E](#_云台命令枚举) |
| dwSpeed | IN | 云台控制的速度,取值范围[1,9] |

**Remarks**：

* 无需启动实况预览

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZControl](#_云台控制操作_1)

### 获取、设置云台守望信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取、设置指定通道的云台守望信息(不需要启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChannelID | IN | 通道ID |
| dwPTZGuardCmd | IN | 云台守望配置命令。参考枚举[NETDEV\_PTZ\_GUARD\_CMD\_E](#_云台守望配置命令) |
| pstPTZGuardInfo | OUT | 拉框缩放结构体 |

**Remarks**：

* 无需启动实况预览

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZHomePosition](#_云台看守位操作)，[NETDEV\_PTZHomePosition\_Other](#_云台看守位操作_1)

### 云台看守位操作

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZHomePosition  (  LPVOID lpRealHandle,  INT32 dwPTZHomePositionCmd  ); |

**接口描述：**

云台看守位操作(需要启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpRealHandle | IN | 实时预览句柄 |
| dwPTZHomePositionCmd | IN | 操作云台看守位命令,参考  枚举[NETDEV\_PTZ\_HOMEPOSITIONCMD\_E](#_云台看守位操作命令) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZHomePosition\_Other](#_云台看守位操作_1)

### 无预览云台看守位操作

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZHomePosition\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwPTZHomePositionCmd  ); |

**接口描述：**

云台看守位操作(不需要启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| dwPTZHomePositionCmd | IN | 操作云台看守位命令。参考[NETDEV\_PTZ\_HOMEPOSITIONCMD\_E](#_云台看守位操作命令) |

**Remarks**：

* 无需启动实况预览

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZHomePosition](#_云台看守位操作)

### 云台绝对坐标转动

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZAbsoluteMove  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_ABSOLUTE\_MOVE\_S](#_绝对坐标移动) pstAbsoluteMove  ); |

**接口描述：**

云台绝对坐标转动

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| pstAbsoluteMove | IN | 云台绝对坐标转动结构体信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

### 获取云台状态

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZGetStatus  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_STATUS\_S](#_云台状态) pstPTZStaus  ); |

**接口描述：**

获取云台状态

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| pstPTZStaus | OUT | 云台状态 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取云台轨迹巡航路径

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取云台轨迹巡航路径

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| pstTrackCruiseInfo | OUT | 巡航路径列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZGetTrackCruise\_V30](#_获取云台轨迹巡航路径)

### 获取云台轨迹巡航路径（包含循环轨迹ID）

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取云台轨迹巡航路径

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChannelID | IN | 通道号 |
| pstTrackCruiseList | OUT | 巡航路径列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZGetTrackCruise](#_获取云台轨迹巡航路径_1)

### 获取指定通道的巡航状态

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取指定通道的巡航状态

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChannelID | IN | 通道ID |
| pstTrackStatus | OUT | 巡航状态信息结构体 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 云台轨迹巡航操作

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZTrackCruise  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwPTZTrackCruiseCmd,  CHAR \*pszTrackCruiseName  ); |

**接口描述：**

云台轨迹巡航操作(不需要启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| dwPTZTrackCruiseCmd | IN | 操作云台轨迹巡航命令。参考[NETDEV\_PTZ\_TRACKCMD\_E](#_云台巡航操作枚举) |
| pszTrackCruiseName | INOUT | 轨迹巡航名称,建议长度64字节，需用户自己分配内存 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZTrackCruise\_V30](#_云台轨迹巡航操作)

### 无预览云台轨迹巡航操作

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

云台轨迹巡航操作(不需要启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChannelID | IN | 通道号 |
| dwPTZTrackCruiseCmd | IN | 操作云台轨迹巡航命令。参考[NETDEV\_PTZ\_TRACKCMD\_E](#_云台巡航操作枚举) |
| pstTrackCruiseInfo | IN | 云台轨迹巡航路径信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZTrackCruise](#_云台轨迹巡航操作_1)

### 获取云台预置位列表

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetPTZPresetList  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_ALLPRESETS\_S](#_所有云台预置位结构体) pstPresetList  ); |

**接口描述：**

获取云台预置位列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChannelID | IN | 通道号 |
| pstPresetList | OUT | 预置位列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZPreset](#_云台预置位操作)，[NETDEV\_PTZPreset\_Other](#_云台预置位操作_1)

### 云台预置位操作

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZPreset  (  LPVOID lpPlayHandle,  INT32 dwPTZPresetCmd,  CHAR \*pszPresetName,  INT32 dwPresetID  ); |

**接口描述：**

云台预置位操作(需先启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄 |
| dwPTZPresetCmd | IN | 操作云台预置位命令。参考枚举[NETDEV\_PTZ\_PRESETCMD\_E](#_云台预置位操作命令枚举) |
| pszPresetName | IN | 预置位的名称 |
| dwPresetID | IN | 预置位的序号（从1开始）,最多支持255个预置位 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZPreset\_Other](#_云台预置位操作_1)

### 无预览云台预置位操作

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PTZPreset\_Other  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwPTZPresetCmd,  CHAR \*pszPresetName,  INT32 dwPresetID  ); |

**接口描述：**

云台预置位操作(不需先启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| dwPTZPresetCmd | IN | 操作云台预置位命令。参考枚举[NETDEV\_PTZ\_PRESETCMD\_E](#_云台预置位操作命令枚举) |
| pszPresetName | IN | 预置位的名称 |
| dwPresetID | IN | 预置位的序号（从1开始）,最多支持255个预置位 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZPreset](#_云台预置位操作)

### 云台预置位巡航操作

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

云台预置位巡航操作(需先启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 实时预览句柄 |
| dwPTZCruiseCmd | IN | 操作云台巡航命令。参考枚举[NETDEV\_PTZ\_CRUISECMD\_E](#_云台巡航操作枚举_1) |
| pstCruiseInfo | IN | 云台巡航路径信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZCruise\_Other](#_云台预置位巡航操作)

### 无预览云台预置位巡航操作

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

云台预置位巡航操作(无需先启动预览)

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| dwPTZCruiseCmd | IN | 操作云台巡航命令。参考枚举[NETDEV\_PTZ\_CRUISECMD\_E](#_云台巡航操作枚举_1) |
| pstCruiseInfo | IN | 云台巡航路径信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_PTZCruise](#_云台预置位巡航操作_1)

### 获取云台预置位巡航路径

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取云台预置位巡航路径

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| pstCruiseList | OUT | 巡航路径列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 云台标定

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

云台标定

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| pstOrientationInfo | IN | 云台方位信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取云台镜头当前视角值

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取指定视频输入通道的云台镜头当前视角值

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChannelID | IN | 通道ID |
| pstLensAngleInfo | OUT | 云台镜头当前视场角 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取云台的经纬度信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetPTZAbsolutePTInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_PT\_POSITION\_INFO\_S](#_云台经纬度信息单位) pstPTPositionInfo  ); |

**接口描述：**

获取云台的经纬度信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| pstPTPositionInfo | OUT | 云台位置信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SetPTZAbsolutePTInfo](#_设置云台的经纬度信息)

### 设置云台的经纬度信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPTZAbsolutePTInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_PTZ\_PT\_POSITION\_INFO\_S](#_云台经纬度信息单位) pstPTPositionInfo  ); |

**接口描述：**

设置云台的经纬度信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| pstPTPositionInfo | IN | 云台位置信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetPTZAbsolutePTInfo](#_获取云台的经纬度信息)

### 获取云台当前变倍倍数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetPTZAbsoluteZoomInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  FLOAT \*pfZoomRatio  ); |

**接口描述：**

获取云台当前变倍倍数

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| pfZoomRatio | OUT | 云台位置信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SetPTZAbsoluteZoomInfo](#_设置云台当前变倍倍数)

### 设置云台当前变倍倍数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPTZAbsoluteZoomInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  FLOAT fZoomRatio  ); |

**接口描述：**

设置云台当前变倍倍数

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| fZoomRatio | IN | 云台位置信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetPTZAbsoluteZoomInfo](#_获取云台当前变倍倍数)

## 系统配置

### 开启/关闭Telnet

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_EnableTelnet  (  LPVOID lpUserID,  BOOL bEnable  ); |

**接口描述：**

开启/关闭Telnet

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| bEnable | IN | telnet状态 0：不使能 1：使能 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 导出配置文件

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetConfigFile  (  LPVOID lpUserID,  CHAR\* pszConfigPath  ); |

**接口描述：**

导出配置文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| pszConfigPath | IN | 配置文件路径（包含文件名称,后缀名为tgz） |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SetConfigFile](#_设置配置文件业务)

### 导入配置文件

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetConfigFile  (  LPVOID lpUserID,  CHAR\* pszConfigPath  ); |

**接口描述：**

导入配置文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| pszConfigPath | IN | 配置文件路径（包含文件名称,命名格式：设备型号\_IP地址\_config.tgz, 如：HIC5621E-L-U\_192.168.3.112\_config.tgz） |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetConfigFile](#_导出配置文件业务)

### 获取设备的配置信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDevConfig  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwCommand,  LPVOID lpOutBuffer,  INT32 dwOutBufferSize,  INT32 \*pdwBytesReturned  ); |

**接口描述：**

获取设备的配置信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| dwCommand | IN | 设备配置命令。参见# [NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令) |
| lpOutBuffer | INOUT | 接收数据的缓冲指针 |
| dwOutBufferSize | OUT | 接收数据的缓冲长度(以字节为单位),不能为0 |
| pdwBytesReturned | OUT | 实际收到的数据长度指针,不能为NULL |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SetDevConfig](#_设置设备的配置信息)

### 设置设备的配置信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetDevConfig  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwCommand,  LPVOID lpInBuffer,  INT32 dwInBufferSize  ); |

**接口描述：**

设置设备的配置信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| dwCommand | IN | 设备配置命令。参见#[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令) |
| lpInBuffer | IN | 输入数据的缓冲指针 |
| dwInBufferSize | IN | 输入数据的缓冲长度(以字节为单位) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetDevConfig](#_获取设备的配置信息)

### 查询硬盘模式

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDiskMode  (  LPVOID lpUserID,  UINT32 \* pudwDiskMode  )； |

**接口描述：**

查询硬盘模式

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| pudwDiskMode | OUT | 硬盘模式。参见枚举[NETDEV\_DISK\_MODE\_TYPE\_E](#_硬盘模式) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SetDiskMode](#_设置硬盘模式)

### 设置硬盘模式

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetDiskMode  (  LPVOID lpUserID,  UINT32 udwDiskMode  ); |

**接口描述：**

设置硬盘模式

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| udwDiskMode | IN | 硬盘模式。参见枚举[NETDEV\_DISK\_MODE\_TYPE\_E](#_硬盘模式) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetDiskMode](#_查询硬盘模式)

### 获取设备系统时间配置

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetSystemTimeCfg  (  LPVOID lpUserID,  [LPNETDEV\_TIME\_CFG\_S](#_时间配置_3) pstSystemTimeInfo  ); |

**接口描述：**

获取设备系统时间配置

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstSystemTimeInfo | OUT | 时间配置信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SetSystemTimeCfg](#_设置设备系统时间配置)

### 设置设备系统时间配置

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetSystemTimeCfg  (  LPVOID lpUserID,  [LPNETDEV\_TIME\_CFG\_S](#_时间配置_3) pstSystemTimeInfo  ); |

**接口描述：**

设置设备系统时间

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstSystemTimeInfo | IN | 时间配置信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetSystemTimeCfg](#_获取设备系统时间配置)

### 无需登录修改设备IP地址

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyDeviceAddr  (  [LPNETDEV\_DEV\_ADDR\_INFO\_S](#_设备地址信息结构体) pstDevAddrInfo  ) |

**接口描述：**

修改设备IP地址（无需登录）

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| pstDevAddrInfo | IN | 要修改的设备地址信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 修改设备名称

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyDeviceName  (  LPVOID lpUserID,  CHAR \*pszDeviceName  ); |

**接口描述：**

修改设备名称

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pszDeviceName | IN | 设备名称 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取设备能力集

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDeviceCapability  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwCommand,  LPVOID lpOutBuffer,  INT32 dwOutBufferSize,  INT32 \*pdwBytesReturned  ); |

**接口描述：**

获取设备能力集

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| dwCommand | IN | 设备能力类型指令。参见[NETDEV\_CAPABILITY\_COMMOND\_E](#_能力集命令) |
| lpOutBuffer | OUT | 接收数据的缓冲指针 |
| dwOutBufferSize | OUT | 接收数据的缓冲长度(以字节为单位) |
| pdwBytesReturned | OUT | 实际收到的数据长度指针 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 通过设备类型查询设备列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindDevList  (  LPVOID lpUserID,  INT32 dwDevType  ); |

**接口描述：**

通过设备类型查询设备列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwDevType | IN | 设备类型参见#[NETDEV\_DEVICE\_MAIN\_TYPE\_E](#_设备类型枚举定义) |

**Return Values：**

查询句柄,返回0表示失败，其他值作为[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)等函数的参数。

**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)

### 逐个获取查找到的设备信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextDevInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1) pstDevBasicInfo  ); |

**接口描述：**

通过设备类型查询设备列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |
| pstDevBasicInfo | OUT | 保存设备详细信息的指针 |

**Return Values：**

TRUE表示成功，其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因

**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)

### 关闭查找设备信息，释放资源

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseDevInfo(LPVOID lpFindHandle); |

**接口描述：**

关闭查找设备信息，释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 文件查找句柄 |

**Return Values：**

TRUE表示成功，其他表示失败

**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)

### 获取通道设备基本信息列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindDevChlBasicInfoList(LPVOID lpUserID); |

**接口描述：**

获取通道设备基本信息列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |

**Return Values：**

业务号,返回表示失败,其他值作为[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)函数的参数

**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)

### 逐个获取通道设备基本信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextDevChlBasicInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_DEVICE\_CHL\_INFO\_S](#_设备通道基本信息) pstDeviceInfo  ); |

**接口描述：**

逐个获取通道设备基本信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查询Handle |
| pstDeviceInfo | OUT | 设备信息结构体指针 |

**Return Values：**

TRUE表示成功,其他表示失败

**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)

### 关闭获取通道设备基本信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseDevChlBasicInfo(IN LPVOID lpFindHandle); |

**接口描述：**

关闭获取通道设备基本信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查询Handle |

**Return Values：**

TRUE表示成功,其他表示失败

**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)

### 通过设备ID或通道类型查询通道信息列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindDevChnList  (  LPVOID lpUserID,  INT32 dwDevID,  INT32 dwChnType  ); |

**接口描述：**

通过设备ID或通道类型查询通道信息列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwDevID | IN | 设备ID |
| dwChnType | IN | 通道类型，参见# [NETDEV\_CHN\_TYPE\_E](#_通道类型) |

**Return Values：**

查询句柄,返回表示失败，其他值作为[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)等函数的参数

**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)

### 逐个获取查找到的设备通道信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextDevChn  (  LPVOID lpFindHandle,  LPVOID lpOutBuffer,  INT32 dwOutBufferSize,  INT32 \*pdwBytesReturned  ); |

**接口描述：**

逐个获取查找到的设备通道信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |
| lpOutBuffer | OUT | 接收数据的缓冲指针 |
| dwOutBufferSize | IN | 接收数据的缓冲长度(以字节为单位)，不能为 |
| pdwBytesReturned | OUT | 实际收到的数据长度指针，不能为NUL |

**Return Values：**

TRUE表示成功，其他表示失败

**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)

### 关闭查找设备通道信息，释放资源

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseDevChn(LPVOID lpFindHandle); |

**接口描述：**

关闭查找设备通道信息，释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |

**Return Values：**

TRUE表示成功，其他表示失败

**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)

### 根据通道类型和通道ID获取通道详细信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetChnDetailByChnType  (  LPVOID lpUserID,  INT32 dwChnID,  INT32 dwChnType,  LPVOID lpOutBuffer,  INT32 dwOutBufferSize,  INT32 \*pdwBytesReturned  ); |

**接口描述：**

根据通道类型和通道ID获取通道详细信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChnID | IN | 通道ID |
| dwChnType | IN | 通道类型,参见# [NETDEV\_CHN\_TYPE\_E](#_通道类型) |
| lpOutBuffer | INOUT | 接收数据的缓冲指针 |
| dwOutBufferSize | IN | 接收数据的缓冲长度(以字节为单位)，不能为0 |
| pdwBytesReturned | OUT | 实际收到的数据长度指针，不能为NULL |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 根据通道类型和通道ID设置报警输出通道的信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetChnDetailByChnType  (  LPVOID lpUserID,  INT32 dwChnID,  INT32 dwChnType,  LPVOID lpOutBuffer,  INT32 dwOutBufferSize  ); |

**接口描述：**

根据通道类型和通道ID设置报警输出通道的信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChnID | IN | 通道ID |
| dwChnType | IN | 通道类型,参见# [NETDEV\_CHN\_TYPE\_E](#_通道类型) |
| lpOutBuffer | IN | 接收数据的缓冲指针 |
| dwOutBufferSize | IN | 接收数据的缓冲长度(以字节为单位)，不能为0 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetChnDetailByChnType](#_根据通道类型和通道ID获取通道详细信息)

### 获取通道类型

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetChnType  (  LPVOID lpUserID,  INT32 dwChnID,  INT32 \*pdwChnType  ); |

**接口描述：**

获取通道类型

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChnID | IN | 通道ID |
| pdwChnType | OUT | 设备通道类型参见# [NETDEV\_CHN\_TYPE\_E](#_通道类型) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取通道设备基本信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDevChlBasicInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_DEVICE\_CHL\_INFO\_S](#_设备通道基本信息) pstDeviceChlInfo  ); |

**接口描述：**

获取通道设备基本信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChannelID | IN | 通道ID |
| pstDeviceChlInfo | OUT | 设备基本信息请 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取设备基础信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDeviceBaseInfo  (  LPVOID lpUserID,  [LPNETDEV\_DEVICE\_BASE\_INFO\_S](#_设备基础信息) pstDeviceInfo  ); |

**接口描述：**

获取设备信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| pstDeviceInfo | OUT | 设备信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取设备信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDeviceInfo  (  LPVOID lpUserID ,  [LPNETDEV\_DEVICE\_INFO\_S](#_设备信息) pstDevInfo  ); |

**接口描述：**

获取设备信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstDevInfo | OUT | 设备信息结构体指针 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 查询设备详细信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDeviceInfo\_V30  (  LPVOID lpUserID,  INT32 dwDevID,  [LPNETDEV\_DEV\_INFO\_V30\_S](#_设备详细信息) pstDevDetailInfo  ); |

**接口描述：**

查询设备详细信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwDevID | IN | 设备ID |
| pstDevDetailInfo | OUT | 设备详细信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取设备定位信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetGeolocationInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_GEOLACATION\_INFO\_S](#_定位信息) pstGPSInfo  ); |

**接口描述：**

获取设备定位信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChannelID | IN | 通道号 |
| pstGPSInfo | OUT | 定位信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取wifi sniffer MAC地址信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetWifiSnifferMacList  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_WIFISNIFFER\_MAC\_LIST\_S](#_WIFISnifferMac数组信息) pstMACList  ); |

**接口描述：**

获取wifi sniffer MAC地址信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| dwChannelID | IN | 通道号 |
| pstMACList | OUT | Wifi sniffer Mac地址数组 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 查询视频通道信息列表

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_QueryVideoChlDetailList  (  LPVOID lpUserID,  INT32 \*pdwChlCount,  [LPNETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_S](#_视频通道详细信息) pstVideoChlList  ); |

**接口描述：**

查询视频通道信息列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pdwChlCount | INOUT | 通道数 |
| pdwChlCount | OUT | 通道能力集列表 |

**Return Values：**

TRUE表示成功，其他表示失败

**See Also：**

[NETDEV\_QueryVideoChlDetailListEx](#_查询视频通道信息列表)、[NETDEV\_QueryVideoChlInfo](#_查询视频通道信息)

### 查询视频通道信息列表

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

查询视频通道信息列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pdwChlCount | INOUT | 通道数 |
| pstVideoChlList | OUT | 通道能力集列表 |

**Return Values：**

TRUE表示成功，其他表示失败

**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)

### 查询视频通道信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_QueryVideoChlInfo  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_S](#_视频通道详细信息) pstVideoChlInfo  ); |

**接口描述：**

查询视频通道信息列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| pstVideoChlInfo | OUT | 视频通道信息 |

**Return Values：**

TRUE表示成功，其他表示失败

**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)

### 导入音频文件

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ImportAudioFile  (  LPVOID lpUserID,  CHAR\* pszFilePath  ); |

**接口描述：**

导入音频文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pszFilePath | IN | 音频文件路径 导入音频须为PCM格式，文件大小不得超过100K |

**Return Values：**

TRUE表示成功，其他表示失败，获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_DeleteAudioFile](#_删除音频文件)

### 删除音频文件

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteAudioFile  (  LPVOID lpUserID,  UINT32 udwID  ); |

**接口描述：**

删除音频文件

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwID | IN | 音频文件ID |

**Return Values：**

TRUE表示成功，其他表示失败，获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_ImportAudioFile](#_导入音频文件)

### 获取音频文件

#### 获取所有的音频文件信息

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindAudioFileList  (  LPVOID lpUserID,  UINT32\* pudwNum  ); |

**接口描述：**

获取所有的音频文件信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pudwNum | OUT | 音频文件个数 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextAudioFileInfo](#_逐个查询获取到的音频文件信息)接口音频文件信息；
* 获取音频文件信息结束后必须调用[NETDEV\_FindCloseAudioFileList](#_关闭查找，释放资源_1)接口，以释放资源,关闭查找；

#### 逐个查询获取到的音频文件信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextAudioFileInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_AUDIO\_FILE\_INFO\_S](#_音频文件信息) pstAudioFileInfo  ); |

**接口描述：**

逐个查询获取到的音频文件信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |
| pstAudioFileInfo | OUT | 音频文件信息 |

**Return Values：**

TRUE表示成功，其他表示失败，获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindAudioFileList](#_获取所有的音频文件信息)、[NETDEV\_FindCloseAudioFileList](#_关闭查找，释放资源_1)接口配套使用；
* 此接口需多次调用，逐个获取组织通道信息；
* 获取音频文件信息结束后必须调用[NETDEV\_FindCloseAudioFileList](#_关闭查找，释放资源_1)接口，以释放资源,关闭查找；

#### 关闭查找音频文件信息，释放资源

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseAudioFileList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭查找音频文件信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 获取组织通道信息结束后必须调用[NETDEV\_FindCloseAudioFileList](#_关闭查找，释放资源_1)接口，以释放资源,关闭查找；

### 查询音频输出参数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetAudioOutputCfg  (  LPVOID lpUserID,  UINT32 udwChannelID,  [LPNETDEV\_AUDIO\_OUTPUT\_CFG\_S](#_音频输出参数) pstAudioOutputCfg  ); |

**接口描述：**

查询音频输出参数

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwChannelID | IN | 通道ID |
| pstAudioOutputCfg | OUT | 音频输出参数 |

**Return Values：**

TRUE表示成功，其他表示失败，获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 设置音频输出参数

|  |
| --- |
| BOOL STDCALL NETDEV\_SetAudioOutputCfg  (  LPVOID lpUserID,  UINT32 udwChannelID,  [LPNETDEV\_AUDIO\_OUTPUT\_CFG\_S](#_音频输出参数) pstAudioOutputCfg  ); |

**接口描述：**

设置音频输出参数

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwChannelID | IN | 通道ID |
| pstAudioOutputCfg | IN | 音频输出参数 |

**Return Values：**

TRUE表示成功，其他表示失败，获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

## 告警抓图

### 获取告警联动抓图URL

#### 获取告警联动抓图URL列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindAlarmSnapShotURL  (  LPVOID lpUserID,  [LPNETDEV\_ALARM\_SNAPSHOT\_COND\_S](#_告警抓图URL查找条件) pstAlarmSnapShotCond  ); |

**接口描述：**

获取告警联动抓图URL列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstAlarmSnapShotCond | IN | 告警抓图获取条件 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextAlarmSnapShotURL](#_逐个查找获取到的URL信息)接口获取告警抓图URL；
* 获取告警抓图URL结束后必须调用[NETDEV\_FindCloseAlarmSnapShotURL](#_关闭告警联动抓图URL列表查找)接口，以释放资源,关闭查找；

**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)

#### 逐个查找获取到的告警图片信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextAlarmSnapShotURL  (  LPVOID lpFindHandle,  [LPNETDEV\_ALARM\_SNAPSHOT\_PIC\_S](#_告警抓拍图片信息) pstAlarmPicInfo  ); |

**接口描述：**

逐个获取查找到的告警抓图信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_FindAlarmSnapShotURL](#_获取告警联动抓图URL列表)接口的返回值 |
| pstAlarmPicInfo | OUT | 告警抓图图片信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 此接口需多次调用，逐个获取告警抓图URL信息；
* 获取告警抓图URL结束后必须调用[NETDEV\_FindCloseAlarmSnapShotURL](#_关闭告警联动抓图URL列表查找)接口，以释放资源,关闭查找；

**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)

#### 关闭告警联动抓图URL列表查找

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseAlarmSnapShotURL  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭告警抓图URL查找,释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_FindAlarmSnapShotURL](#_获取告警联动抓图URL列表)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 获取告警抓图URL结束后必须调用[NETDEV\_FindCloseAlarmSnapShotURL](#_关闭告警联动抓图URL列表查找)接口，以释放资源,关闭查找；

**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)

### 保存告警抓图

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SaveSnapShotFile  (  LPVOID lpUserID,  [LPNETDEV\_PIC\_FILE\_INFO\_S](#_抓拍图片信息结构体) pstPicFileInfo  ); |

**接口描述：**

保存告警抓图

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstPicFileInfo | IN | 告警图片信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

## 告警消息

### 注册回调函数,接收报警消息等

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetAlarmCallBack\_V30  (  LPVOID lpUserID,  NETDEV\_AlarmMessCallBack\_PF\_V30 cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册回调函数,接收报警消息等

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| cbAlarmMessCallBack | IN | 回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| pstReportInfo | IN | 上报信息 |
| lpBuf | IN | 缓冲区指针 |
| dwBufLen | IN | 告警信息对应结构体的长度 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功，其他表示失败

**Remarks**：

* 若关闭回调函数,将第二个参数置为NULL；

### 主动获取设备告警信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_PullAlarm  (  LPVOID lpUserID,  INT32 dwPullWaitTime,  INT32 \*pdwListCnt,  [LPNETDEV\_PULLALARM\_INFO\_S](#_拉告警信息) pstPullAlarmList  ); |

**接口描述：**

主动获取设备告警信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwPullWaitTime | IN | 告警等待时间,单位s |
| pdwListCnt | INOUT | 告警最大数量,入参数时表示告警信息列表大小,出参时表示实际告警数量 |
| pstPullAlarmList | OUT | 告警信息列表,此内存需预先分配 |

**Return Values：**

TRUE表示成功，其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

## 客流量

### 获取客流量统计信息列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindTrafficStatisticInfoList  (  LPVOID lpUserID,  UINT32 udwSearchID  ); |

**接口描述：**

获取客流量统计信息列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| udwSearchID | IN | 客流量查询ID |

**Return Values：**

查询句柄(作为[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表示失败

**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)

### 逐个查询获取到的客流量信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextTrafficStatisticInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_TRAFFIC\_STATISTICS\_INFO\_S](#_客流量统计) pstStatisticInfo  ); |

**接口描述：**

逐个查询获取到的客流量信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |
| pstStatisticInfo | OUT | 客流量查询信息 |

**Return Values：**

TRUE表示成功，其他表示失败

**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)

### 关闭查找，释放资源

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseTrafficStatisticInfo(IN LPVOID lpFindHandle) |

**接口描述：**

关闭查找，释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |

**Return Values：**

TRUE表示成功，其他表示失败

**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)

### 获取客流量统计

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取客流量统计

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstStatisticCond | IN | 客流量统计命令 |
| pstTrafficStatistic | OUT | 客流量统计 |

**Return Values：**

TRUE表示成功，其他表示失败

**See Also：**

### 清零客流量人数统计

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ResetPassengerFlow  (  LPVOID lpUserID,  INT32 dwChannelID  )； |

**接口描述：**

清零客流量人数统计

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |

**Return Values：**

TRUE表示成功，其他表示失败

**See Also：**

### 获取客流量上报时间间隔

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetTrafficStatisticReportTime  (  LPVOID lpUserID,  INT32 dwChannelID,  [LPNETDEV\_TRAFFIC\_STATISTICS\_REPORT\_TIME\_S](#_客流量上报时间间隔) pstReportTimeValue  ); |

**接口描述：**

获取客流量上报时间间隔

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwChannelID | IN | 通道号 |
| pstReportTimeValue | OUT | 客流量上报间隔时间 |

**Return Values：**

TRUE表示成功，其他表示失败

**See Also：**

### 开始客流量统计查询

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StartTrafficStatistic  (  LPVOID lpUserID,  [LPNETDEV\_TRAFFIC\_STATISTICS\_COND\_S](#_客流量统计命令) pstStatisticCond,  UINT32\* pudwSearchID  ); |

**接口描述：**

开始客流量统计查询

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| pstStatisticCond | IN | 客流量统计结构体 |
| pudwSearchID | OUT | 客流量查询ID |

**Return Values：**

TRUE表示成功，其他表示失败

**See Also：**

[NETDEV\_StopTrafficStatistic](#_停止客流量查询)

### 停止客流量查询

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopTrafficStatistic  (  LPVOID lpUserID,  UINT32 udwSearchID  ); |

**接口描述：**

**停止客流量查询**

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| udwSearchID | IN | 客流量查询ID |

**Return Values：**

TRUE表示成功，其他表示失败

**See Also：**

[NETDEV\_StartTrafficStatistic](#_开始客流量统计查询)

### 多通道开始客流量统计查询

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

多通道开始客流量统计查询

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| pstStatisticCond | IN | 客流量统计结构体 |
| pudwSearchID | OUT | 客流量查询ID |

**Return Values：**

TRUE表示成功，其他表示失败

**See Also：**

### 获取客流量统计进度

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetTrafficStatisticProgress  (  LPVOID lpUserID,  UINT32 udwSearchID,  UINT32\* pudwProgress  ); |

**接口描述：**

获取客流量统计进度

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录ID |
| udwSearchID | IN | 客流量查询ID |
| pudwProgress | OUT | 客流量统计进度 |

**Return Values：**

TRUE表示成功，其他表示失败

**See Also：**

### 注册客流量统计上报回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPassengerFlowStatisticCallBack  (  LPVOID lpUserID,  NETDEV\_PassengerFlowStatisticCallBack\_PF cbPassengerFlowStatisticCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册客流量统计上报回调函数

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| cbPassengerFlowStatisticCallBack | IN | 需要注册的回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstPassengerFlowData | IN | 客流量数据 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功，其他表示失败

**Remarks**：

* 若关闭回调函数,将第二个参数置为NULL；

**See Also：**

## 布控任务

### 新增指定的人员信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

新增指定的人员信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwPersonLibID | IN | 人员库ID |
| pstPersonInfoList | IN | 人员信息列表 |
| pstPersonResultList | OUT | 人员信息结果列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 删除指定的人员信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeletePersonInfo  (  LPVOID lpUserID,  UINT32 udwPersonLibID,  UINT32 udwPersonID,  UINT32 udwLastChange  ); |

**接口描述：**

删除指定的人员信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwPersonLibID | IN | 人员库ID |
| udwPersonID | IN | 人员ID |
| udwLastChange | IN | 最后修改时间 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 新增单个人脸布控任务

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

新增单个人脸布控任务

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstMonitorInfo | INOUT | 布控任务信息，成功返回布控任务序号 |
| pstMonitorResultInfo | INOUT | 添加布控返回的布控信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。pstMonitorResultInfo->udwChannelNum不应小于pstMonitorInfo stMonitorRuleInfo.udwChannelNum。

**See Also：**

[NETDEV\_DeletePersonMonitorInfo](#_删除单个人脸布控任务)

### 删除单个人脸布控任务

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ DeletePersonMonitorInfo  (  LPVOID lpUserID,  UINT32 udwID  ); |

**接口描述：**

删除单个人脸布控任务

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwID | IN | 布控ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_AddPersonMonitorInfo](#_新增单个人脸布控任务)

### 批量删除人脸布控任务

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量删除人脸布控任务

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstResultList | INOUT | 返回信息列表，需要输入布控要删除的所有布控ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

### 创建人员库信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_CreatePersonLibInfo  (  LPVOID lpUserID,  [LPNETDEV\_LIB\_INFO\_S](#_人员库信息结构体) pstPersonLibInfo,  UINT32 \*pudwID  ); |

**接口描述：**

创建人员库信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstPersonLibInfo | IN | 人员库信息 |
| pudwID | OUT | 创建库生成的库ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_ ModifyPersonLibInfo](#_修改人员库信息)、[NETDEV\_DeletePersonLibInfo](#_删除指定的人员库)

### 修改人员库信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyPersonLibInfo  (  LPVOID lpUserID,  [LPNETDEV\_PERSON\_LIB\_LIST\_S](#_人员库信息列表结构体) pstPersonLibList  ); |

**接口描述：**

修改人员库信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstPersonLibList | IN | 人员库信息列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_CreatePersonLibInfo](#_创建人员库信息)、[NETDEV\_DeletePersonLibInfo](#_删除指定的人员库)

### 删除指定的人员库

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeletePersonLibInfo  (  LPVOID lpUserID,  UINT32 udwPersonLibID,  [LPNETDEV\_DELETE\_DB\_FLAG\_INFO\_S](#_删除库标志位结构体) pstFlagInfo  ); |

**接口描述：**

删除指定的人员库

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwPersonLibID | IN | 人员库ID |
| pstFlagInfo | IN | 人员库删除标志位 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_ CreatePersonLibInfo](#_创建人员库信息)、[NETDEV\_ ModifyPersonLibInfo](#_修改人员库信息)

### 查询单个人脸布控任务配置信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetPersonMonitorRuleInfo  (  LPVOID lpUserID,  [LPNETDEV\_MONITION\_INFO\_S](#_布控任务信息结构体) pstMonitorInfo  ); |

**接口描述：**

查询单个人脸布控任务配置信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstMonitorInfo | INOUT | 布控任务信息；需输入布控ID，成功返回配置信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。布控任务信息->人脸布控任务配置信息中pudwMonitorChlIDList由上层申请，接口调用失败错误码为：NETDEV\_E\_NEED\_MORE\_MEMORY时通过udwChannelNum判断是否内存申请过小。

**See Also：**

[NETDEV\_ SetPersonMonitorRuleInfo](#_设置单个人脸布控任务配置信息)

### 设置单个人脸布控任务配置信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ SetPersonMonitorRuleInfo  (  LPVOID lpUserID,  [LPNETDEV\_MONITION\_INFO\_S](#_布控任务信息结构体) pstMonitorInfo  ); |

**接口描述：**

设置单个人脸布控任务配置信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstMonitorInfo | IN | 布控任务信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_ GetPersonMonitorRuleInfo](#_查询单个人脸布控任务配置信息)

### 查询所有人员库的容量信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

查询所有人员库的容量信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwTimeOut | IN | 连接超时时间 |
| pstCapacityList | OUT | 所有人员库的容量信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

### 条件查询人员信息

#### 查询人员信息列表

|  |
| --- |
| 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  ); |

**接口描述：**

条件查询人员信息列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwPersonLibID | IN | 人员库ID |
| pstQueryInfo | IN | 人脸信息查询条件 |
| pstQueryResultInfo | OUT | 人脸信息查询返回结果 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextPersonInfo](#_逐个获取查找到的人员信息)、[NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextPersonInfo](#_逐个获取查找到的人员信息)、[NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextPersonInfo](#_逐个获取查找到的人员信息)接口人员信息；
* 获取人员信息结束后必须调用[NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找)接口，以释放资源,关闭查找；

**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)

#### 逐个获取查找到的人员信息

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextPersonInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_PERSON\_INFO\_S](#_人员信息结构体) pstPersonInfo  ); |

**接口描述：**

逐个获取查找到的人员信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 文件查找句柄 |
| pstPersonInfo | OUT | 人员信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindPersonInfoList](#_查询人员信息列表)、[NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找)接口配套使用
* 此接口需多次调用，逐个获取人员信息；
* 获取人员信息结束后必须调用[NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找)接口，以释放资源,关闭查找；

**See Also：**

[NETDEV\_FindPersonInfoList](#_查询人员信息列表)、[NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找)

#### 关闭人员信息查找

|  |
| --- |
| BOOL STDCALL NETDEV\_FindClosePersonInfoList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭人员信息查找,释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 文件查找句柄，[NETDEV\_FindPersonInfoList](#_查询人员信息列表)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindPersonInfoList](#_查询人员信息列表)、[NETDEV\_FindNextPersonInfo](#_逐个获取查找到的人员信息)接口配套使用
* 获取人员信息结束后必须调用[NETDEV\_FindClosePersonInfoList](#_关闭人员信息查找)接口，以释放资源,关闭查找；

**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)

### 查询所有已创建的人员库信息

#### 查询所有已创建的人员库信息

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindPersonLibList  (  LPVOID lpUserID  ); |

**接口描述：**

查询所有已创建的人员库信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextPersonLibInfo](#_逐个获取查找到的人脸库信息)接口获取人脸库信息；
* 获取人脸库信息结束后必须调用[NETDEV\_FindClosePersonLibList](#_关闭人脸库信息查找)接口，以释放资源,关闭查找；

**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)

#### 逐个获取查找到的人脸库信息

|  |
| --- |
| BOOL STDCALL NETDEV\_ FindNextPersonLibInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_LIB\_INFO\_S](#_人员库信息结构体) pstPersonLibInfo  ); |

**接口描述：**

逐个获取查找到的人脸库信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |
| pstPersonLibInfo | OUT | 人脸库信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 此接口需多次调用，逐个获取人脸库信息；
* 获取人脸库信息结束后必须调用[NETDEV\_FindClosePersonLibList](#_关闭人脸库信息查找)接口，以释放资源,关闭查找；

**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)

#### 关闭人脸库信息查找

|  |
| --- |
| BOOL STDCALL NETDEV\_FindClosePersonLibList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭人员信息查找,释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindPersonInfoList](#_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)接口配套使用
* 获取人脸库信息结束后必须调用[NETDEV\_FindClosePersonInfoList](#_关闭人脸库信息查找)接口，以释放资源,关闭查找；

**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)

### 查询所有人脸布控任务

#### 查询所有人脸布控任务列表

|  |
| --- |
| 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  ); |

**接口描述：**

查询所有人脸布控任务列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwChannelID | IN | 通道ID，仅NVR查询通道布控信息时使用 |
| pstQueryInfo | IN | 查询条件，仅NVR支持 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextPersonMonitorInfo](#_逐个获取查找到的布控任务信息)接口布控任务信息；
* 获取布控任务信息结束后必须调用[NETDEV\_FindClosePersonMonitorList](#_关闭布控任务信息查找)接口，以释放资源,关闭查找；

**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)

#### 逐个获取查找到的布控任务信息

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextPersonMonitorInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_MONITION\_INFO\_S](#_布控任务信息结构体) pstMonitorInfo  ); |

**接口描述：**

逐个获取查找到的布控任务信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |
| pstMonitorInfo | OUT | 布控任务信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 此接口需多次调用，逐个获取布控任务信息；
* 获取布控任务信息结束后必须调用[NETDEV\_FindClosePersonMonitorList](#_关闭布控任务信息查找)接口，以释放资源,关闭查找；

**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)

#### 关闭布控任务信息查找

|  |
| --- |
| BOOL STDCALL NETDEV\_FindClosePersonMonitorList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭布控任务信息查找,释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 获取布控任务信息结束后必须调用[NETDEV\_FindClosePersonMonitorList](#_关闭布控任务信息查找)接口，以释放资源,关闭查找；

**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)

### 新增单个车辆库信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddVehicleLibInfo  (  LPVOID lpUserID,  [LPNETDEV\_LIB\_INFO\_S](#_人员库信息结构体) pstVehicleLibInfo  ); |

**接口描述：**

新增单个车辆库信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstVehicleLibInfo | INOUT | 车库信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 删除指定的车辆库信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteVehicleLibInfo  (  LPVOID lpUserID,  UINT32 udwVehicleLibID,  [LPNETDEV\_DELETE\_DB\_FLAG\_INFO\_S](#_删除库标志位结构体) pstDelLibFlag  ); |

**接口描述：**

删除指定的车辆库信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwVehicleLibID | IN | 车辆库ID |
| pstDelLibFlag | IN | 删除库信息的标志位 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 修改指定的车辆库信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyVehicleLibInfo  (  LPVOID lpUserID,  [LPNETDEV\_PERSON\_LIB\_LIST\_S](#_人员库信息列表结构体) pstVehicleLibList  ); |

**接口描述：**

修改指定的车辆库信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstVehicleLibList | IN | 车辆库列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 批量添加车辆成员信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量添加车辆成员信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwLibID | IN | 车辆库ID |
| pstVehicleMemberList | IN | 车辆信息列表 |
| pstResultList | OUT | 批量添加返回结果信息列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_DelVehicleMemberList](#_批量删除车辆成员信息)

### 批量删除车辆成员信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量删除车辆成员信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwLib | IN | 库序号 |
| pstVehicleMemberList | IN | 车辆成员列表 |
| pstBatchList | OUT | 批量操作返回信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_AddVehicleMemberList](#_批量添加车辆成员信息)

### 删除指定车辆成员信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_DelVehicleInfo  (  LPVOID lpUserID,  UINT32 udwLibID,  UINT32 udwVehicleID  ); |

**接口描述：**

删除指定车辆成员信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwLibID | IN | 车辆库ID |
| udwVehicleID | IN | 车辆ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

### 新增单个车辆布控任务

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddVehicleMonitorInfo  (  LPVOID lpUserID,  [LPNETDEV\_MONITION\_INFO\_S](#_布控任务信息结构体) pstMonitorInfo  ); |

**接口描述：**

新增单个车辆布控任务

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstMonitorInfo | INOUT | 布控任务信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_DeleteVehicleMonitorInfo](#_批量删除车辆布控任务)

### 批量删除车辆布控任务

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量删除车辆布控任务

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstBatchList | INOUT | 信息列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_AddVehicleMonitorInfo](#_新增单个车辆布控任务)

### 获取单个车辆成员详细信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetVehicleMemberInfo  (  LPVOID lpUserID,  UINT32 udwVehicleID,  [LPNETDEV\_VEHICLE\_DETAIL\_INFO\_S](#_车辆成员信息) pstVehicleDetailInfo  ); |

**接口描述：**

获取单个车辆成员详细信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwVehicleID | IN | 车辆ID |
| pstVehicleDetailInfo | OUT | 车辆详细信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

### 修改指定车辆库中车辆信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

修改指定车辆库中车辆信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwVehicleLibID | IN | 车辆库ID |
| pstVehicleMemberList | IN | 车辆详细信息 |
| pstResultList | OUT | 批量操作返回信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

### 查询单个车辆布控任务配置信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

查询单个车辆布控任务配置信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwID | IN | 车辆布控ID |
| pstMonitorInfo | OUT | 布控任务信息 输入布控ID,成功时返回配置信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 车辆布控比对照片内存由使用者维护，入参时，需要指定接收图片缓存大小，失败时会返回实际需要的大小
* 对应错误码：NETDEV\_E\_NEED\_MORE\_MEMORY 用户分配内存不够；
* 如果没有布控图片时，布控图片的大小将会被置为0;；

**See Also：**

[NETDEV\_SetVehicleMonitorInfo](#_设置单个车辆布控任务配置信息)

### 设置单个车辆布控任务配置信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

设置单个车辆布控任务配置信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwID | IN | 车辆布控ID |
| pstMonitorInfo | IN | 布控任务信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetVehicleMonitorInfo](#_查询单个车辆布控任务配置信息)

### 向指定的车辆库中批量划归车辆成员

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

向指定的车辆库中批量划归车辆成员

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwVehicleLibID | IN | 车辆库ID |
| pstMemberList | IN | 批量划归车辆成员ID信息 |
| pstBatchResultList | OUT | 批量划归车辆信息返回结果 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_DeleteVehicleLibMember](#_批量取消指定的车辆库中车辆成员划归)

### 批量取消指定的车辆库中车辆成员划归

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量取消指定的车辆库中车辆成员划归

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwVehicleLibID | IN | 车辆库ID |
| pstMemberList | IN | 批量取消车辆成员ID信息 |
| pstBatchResultList | OUT | 批量取消车辆信息返回结果 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_AddVehicleLibMember](#_向指定的车辆库中批量划归车辆成员)

### 查询单个车辆识别记录的车辆图片信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetVehicleRecordImageInfo  (  LPVOID lpUserID,  UINT32 udwRecordID,  [LPNETDEV\_FILE\_INFO\_S](#_文件信息结构体) pstFileInfo  ); |

**接口描述：**

查询单个车辆识别记录的车辆图片信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwRecordID | IN | 车辆识别记录ID |
| pstFileInfo | INOUT | 车辆图片信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

### 查询全部车辆库信息列表

#### 查询全部车辆库信息列表

|  |
| --- |
| LPVOID STDCALL NETDEV\_ FindVehicleLibList  (  LPVOID lpUserID  ); |

**接口描述：**

查询全部车辆库信息列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextVehicleLibInfo](#_逐个获取查找到的车辆库信息)接口获取车辆库信息；
* 获取车辆库信息结束后必须调用[NETDEV\_FindCloseVehicleLibList](#_关闭车辆库信息查找)接口，以释放资源,关闭查找；

**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)

#### 逐个获取查找到的车辆库信息

|  |
| --- |
| BOOL STDCALL NETDEV\_ FindNextVehicleLibInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_LIB\_INFO\_S](#_人员库信息结构体) pstVehicleLibInfo  ); |

**接口描述：**

逐个获取查找到的车辆库信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |
| pstVehicleLibInfo | OUT | 车辆库信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindVehicleLibList](#_查询全部车辆库信息列表)、[NETDEV\_FindCloseVehicleLibList](#_关闭车辆库信息查找)接口配套使用
* 此接口需多次调用，逐个获取车辆库信息；
* 获取车辆库信息结束后必须调用[NETDEV\_FindCloseVehicleLibList](#_关闭车辆库信息查找)接口，以释放资源,关闭查找；

**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)

#### 关闭车辆库信息查找

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseVehicleLibList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭车辆库信息查找,释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 获取车辆库信息结束后必须调用[NETDEV\_FindCloseVehicleLibList](#_关闭车辆库信息查找)接口，以释放资源,关闭查找；

**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)

### 条件查询车辆识别记录的详细信息

#### 条件查询车辆识别记录的详细信息

|  |
| --- |
| 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  ); |

**接口描述：**

条件查询车辆识别记录的详细信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstFindCond | IN | 查询条件 |
| pstResultInfo | OUT | 查询的记录信息 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextVehicleRecordInfo](#_逐个获取查找到的车辆识别记录信息)接口获取车辆识别记录信息；
* 获取车辆识别记录信息结束后必须调用[NETDEV\_FindCloseVehicleRecordList](#_关闭车辆识别记录信息查找)接口，以释放资源,关闭查找；

**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)

#### 逐个获取查找到的车辆识别记录信息

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextVehicleRecordInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_VEHICLE\_RECORD\_INFO\_S](#_车辆识别记录信息结构体) pstRecordInfo  ); |

**接口描述：**

逐个获取查找到的车辆识别记录信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |
| pstRecordInfo | OUT | 车辆识别记录信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 此接口需多次调用，逐个获取车辆识别记录信息；
* 获取车辆识别记录信息结束后必须调用[NETDEV\_FindCloseVehicleRecordList](#_关闭车辆识别记录信息查找)接口，以释放资源,关闭查找；

**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)

#### 关闭车辆识别记录信息查找

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseVehicleRecordList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭车辆识别记录信息查找,释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 获取车辆识别记录信息结束后必须调用[NETDEV\_FindCloseVehicleRecordList](#_关闭车辆识别记录信息查找)接口，以释放资源,关闭查找；

**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)

### 查询车辆识别的所有布控任务

#### 查询车辆识别的所有布控任务

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindVehicleMonitorList  (  LPVOID lpUserID  ); |

**接口描述：**

查询车辆识别的所有布控任务

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextVehicleMonitorInfo](#_逐个获取查找到的车辆布控任务信息)接口获取车辆布控任务信息；
* 获取车辆布控任务信息结束后必须调用[NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找)接口，以释放资源,关闭查找；

**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)

#### 逐个获取查找到的车辆布控任务信息

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextVehicleMonitorInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_MONITION\_INFO\_S](#_布控任务信息结构体) pstVehicleMonitorInfo  ); |

**接口描述：**

逐个获取查找到的车辆布控任务信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |
| pstVehicleMonitorInfo | OUT | 布控任务信息 |

**Return Values：**

TRUE表示成功,其他表示失败。车辆布控比对照片内存由使用者维护，入参时，需要指定接收图片缓存大小，失败时会返回实际需要的大小；如果没有布控图片时，布控图片的大小将会被置为0；图片数据需要取出另存，否则在调用[NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找)接口后内存将被释放；获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 此接口需多次调用，逐个获取车辆布控任务信息；
* 获取车辆布控任务信息结束后必须调用[NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找)接口，以释放资源,关闭查找；

**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)

#### 关闭车辆布控任务信息信息查找

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseVehicleMonitorList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭车辆布控任务信息查找,释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 获取车辆布控任务信息结束后必须调用[NETDEV\_FindCloseVehicleMonitorList](#_关闭车辆布控任务信息信息查找)接口，以释放资源,关闭查找；

**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)

### 条件查询车辆成员详细信息

#### 条件查询车辆成员详细信息

|  |
| --- |
| 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  ); |

**接口描述：**

条件查询车辆成员详细信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwVehicleLibID | IN | 库序号 |
| pstFindCond | IN | 查询条件 |
| pstDBMemberList | OUT | 批量查询返回的基本信息 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextVehicleMemberDetail](#_逐个获取查找到的车辆成员信息)接口获取车辆成员信息；
* 获取车辆成员信息结束后必须调用[NETDEV\_FindCloseVehicleMemberDetail](#_关闭车辆成员信息信息查找)接口，以释放资源,关闭查找；

**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)

#### 逐个获取查找到的车辆成员信息

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextVehicleMemberDetail  (  LPVOID lpFindHandle,  [LPNETDEV\_VEHICLE\_DETAIL\_INFO\_S](#_车辆成员信息) pstVehicleMemberInfo  ); |

**接口描述：**

逐个获取查找到的车辆成员信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |
| pstVehicleMemberInfo | OUT | 车辆成员信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 此接口需多次调用，逐个获取车辆成员信息；
* 获取车辆成员信息结束后必须调用[NETDEV\_FindCloseVehicleMemberDetail](#_关闭车辆成员信息信息查找)接口，以释放资源,关闭查找；

**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)

#### 关闭车辆成员信息信息查找

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseVehicleMemberDetail  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭车辆成员信息查找,释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 获取车辆成员信息结束后必须调用[NETDEV\_FindCloseVehicleMemberDetail](#_关闭车辆成员信息信息查找)接口，以释放资源,关闭查找；

**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)

### 修改指定的人员信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

修改指定的人员信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwPersonLibID | IN | 人员库ID |
| pstPersonInfoList | IN | 人员信息列表 |
| pstPersonResultList | OUT | 人员信息结果列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 批量删除人员信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量删除人员信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| udwPersonLibID | IN | 人脸库ID |
| pstIDList | IN | 人脸成员列表 |
| pstResutList | OUT | 批量操作返回信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

## 数据透传

### 建立透明通道

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_SerialStart  (  LPVOID lpUserID,  LPVOID lpInBuffer,  INT64 ulInBufferSize,  NETDEV\_SerialDataCallBack\_PF cbSerialDataCallBack,  LPVOID lpUserData  ); |

**接口描述：**

建立透明通道；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| lpInBuffer | IN | 输入缓冲区，指向透明通道串口参数,参考[NETDEV\_SERIAL\_START\_S](#_建立透明通道参数结构体) |
| ulInBufferSize | IN | 输入缓冲区大小 |
| cbSerialDataCallBack | IN | 回调函数 |
| lpUserData | IN | 用户数据 |

**Callback Function**

|  |
| --- |
| typedef void(STDCALL \*NETDEV\_SerialDataCallBack\_PF)  (  LPVOID lpSerialHandle,  INT32 dwChannelID,  CHAR\* pRecvDataBuffer,  INT64 ulBufSize,  LPVOID lpUserData  ); |

**Callback Function Parameters**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpSerialHandle | IN | 建立透明通道时返回句柄 |
| dwChannelID | IN | 通道号 |
| pRecvDataBuffer | IN | 存放数据的缓冲区指针 |
| ulBufSize | IN | 数据大小 |
| lpUserData | IN | 用户设置参数 |

**Return Values：**

0表示失败,其他表示返回的透明通道句柄值。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

### 通过透明通道向设备串口发送数据

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SerialSend  (  LPVOID lpSerialHandle,  INT32 dwChannelID,  CHAR\* pSendBuf,  INT64 ulBufSize  ); |

**接口描述：**

通过透明通道向设备串口发送数据

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpSerialHandle | IN | 建立透明通道时返回句柄,[NETDEV\_SerialStart](#_建立透明通道)的返回值 |
| dwChannelID | IN | 串口号 |
| pSendBuf | IN | 发送数据的缓冲区指针 |
| ulBufSize | IN | 缓冲区的大小 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SerialStart](#_建立透明通道)、[NETDEV\_SerialStop](#_断开透明通道)

### 断开透明通道

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SerialStop  (  LPVOID lpSerialHandle  ); |

**接口描述：**

断开透明通道

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpSerialHandle | IN | 建立透明通道时返回句柄,[NETDEV\_SerialStart](#_建立透明通道)的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**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)

## 用户配置

### 创建用户信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_CreateUser  (  LPVOID lpUserID,  [LPNETDEV\_USER\_DETAIL\_INFO\_S](#_用户信息结构体) pstUserDetailInfo  ); |

**接口描述：**

创建用户信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstUserDetailInfo | IN | 用户信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_ModifyUser](#_修改用户信息)、[NETDEV\_DeleteUser](#_删除用户信息)

### 修改用户信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyUser  (  LPVOID lpUserID,  [LPNETDEV\_USER\_MODIFY\_DETAIL\_INFO\_S](#_修改用户详细信息结构体) pstUserInfo  ); |

**接口描述：**

修改用户信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstUserInfo | IN | 用户信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_CreateUser](#_创建用户信息)、[NETDEV\_DeleteUser](#_删除用户信息)

### 删除用户信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteUser  (  LPVOID lpUserID,  CHAR \*pszUserName  ); |

**接口描述：**

删除用户信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pszUserName | IN | 用户名 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_CreateUser](#_创建用户信息)、[NETDEV\_ModifyUser](#_修改用户信息)

### 修改当前用户密码

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ChangePassword  (  LPVOID lpUserID,  CHAR\* pszNewPasswd  ); |

**接口描述：**

修改当前用户密码

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pszNewPasswd | IN | 新密码 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

### 修改指定用户密码

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyPassword  (  LPVOID lpUserID,  [LPNETDEV\_MODIFY\_PASSWORD\_INFO\_S](#_用户密码信息结构体) pstModifyPasswordInfo  ); |

**接口描述：**

修改指定用户密码

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstModifyPasswordInfo | INOUT | 用户密码信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

### 获取设备当前密码

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetDevPassword  (  LPVOID lpUserID,  CHAR \*pszPassword  ); |

**接口描述：**

获取设备当前密码

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pszPassword | OUT | 密码 内存长度要求大于等于NETDEV\_LEN\_64 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

### 获取指定用户全部信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetUserDetailInfo  (  LPVOID lpUserID,  [LPNETDEV\_USER\_DETAIL\_INFO\_S](#_用户信息结构体) pstUserDetailInfo  ); |

**接口描述：**

获取指定用户全部信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstUserDetailInfo | INOUT | 用户信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

### 获取所有用户全部信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetUserDetailList  (  LPVOID lpUserID,  [LPNETDEV\_USER\_DETAIL\_LIST\_S](#_用户信息列表结构体) pstUserDetailList  ); |

**接口描述：**

获取所有用户全部信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstUserDetailList | OUT | 用户信息列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

## 组织管理

### 添加组织

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddOrgInfo  (  LPVOID lpUserID,  [LPNETDEV\_ORG\_INFO\_S](#_组织信息结构体) pstOrgInfo,  INT32 \*dwOrgID  ); |

**接口描述：**

添加组织

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstOrgInfo | IN | 组织信息 |
| dwOrgID | OUT | 组织ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_ModifyOrgInfo](#_修改组织)、[NETDEV\_BatchDeleteOrgInfo](#_批量删除组织)

### 修改组织

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyOrgInfo  (  LPVOID lpUserID,  [LPNETDEV\_ORG\_INFO\_S](#_组织信息结构体) pstOrgInfo  ); |

**接口描述：**

修改组织

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstOrgInfo | IN | 组织信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_AddOrgInfo](#_添加组织)、[NETDEV\_BatchDeleteOrgInfo](#_批量删除组织)

### 批量删除组织

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量删除组织

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstOrgDelInfo | IN | 待删除组织信息 |
| pstOrgDelResultInfo | OUT | 删除组织响应信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_AddOrgInfo](#_添加组织)、[NETDEV\_ModifyOrgInfo](#_修改组织)

### 根据组织ID修改通道

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyOrgChnInfo  (  LPVOID lpUserID,  [LPNETDEV\_ORG\_CHN\_SHORT\_INFO\_S](#_组织下通道信息结构体) pstOrgChnShortInfo  ); |

**接口描述：**

根据组织ID修改通道

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstOrgChnShortInfo | OUT | 组织下通道信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetOrgChnInfo](#_根据组织ID获取通道)

### 根据组织ID获取通道

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

根据组织ID获取通道

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwOrgID | IN | 组织ID |
| pstOrgChnShortInfo | OUT | 组织下通道信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

该接口需要调用两次：第一次获取通道数量 并返回NETDEV\_E\_NEED\_MORE\_MEMORY错误码，第二次根据数量进行动态内存分配后，再次调用此接口获取通道信息。

**See Also：**

[NETDEV\_ModifyOrgChnInfo](#_根据组织ID修改通道)

### 通过组织ID查询通道信息列表

#### 通过组织ID查询通道信息列表

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindOrgChnList  (  LPVOID lpUserID,  INT32 dwOrgID,  INT32 dwChnType  ); |

**接口描述：**

通过组织ID查询通道信息列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| dwOrgID | IN | 组织ID |
| dwChnType | IN | 通道类型，参见#[NETDEV\_CHN\_TYPE\_E](#_通道类型) |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_ FindNextOrgChn](#_逐个获取查找到的组织通道信息)接口获取组织通道信息；
* 获取组织通道信息结束后必须调用[NETDEV\_FindCloseOrgChn](#_关闭组织通道信息查找)接口，以释放资源,关闭查找；

**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)

#### 逐个获取查找到的组织通道信息

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextOrgChn  (  LPVOID lpFindHandle,  [LPNETDEV\_ORG\_CHN\_INFO\_S](#_组织通道信息结构体) pstOrgChnInfo  ); |

**接口描述：**

逐个获取查找到的组织通道信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |
| pstOrgChnInfo | OUT | 组织通道信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 此接口需多次调用，逐个获取组织通道信息；
* 获取组织通道信息结束后必须调用[NETDEV\_FindCloseOrgChn](#_关闭组织通道信息查找)接口，以释放资源,关闭查找；

**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)

#### 关闭组织通道信息查找

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseOrgChn  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭组织通道信息查找,释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用
* 获取组织通道信息结束后必须调用[NETDEV\_FindCloseOrgChn](#_关闭组织通道信息查找)接口，以释放资源,关闭查找；

**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)

### 查询组织信息列表

#### 查询组织信息列表

|  |
| --- |
| 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  ); |

**接口描述：**

查询组织信息列表

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstFindCond | IN | 查找组织信息列表条件信息 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextOrgInfo](#_逐个获取查找到的组织信息)、[NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextOrgInfo](#_逐个获取查找到的组织信息)、[NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_ FindNextOrgInfo](#_逐个获取查找到的组织信息)接口获取组织信息；
* 获取组织信息结束后必须调用[NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找)接口，以释放资源,关闭查找；

**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)

#### 逐个获取查找到的组织信息

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextOrgInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_ORG\_INFO\_S](#_组织信息结构体) pstOrgInfo  ); |

**接口描述：**

逐个获取查找到的组织信息

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |
| pstOrgInfo | OUT | 组织信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindOrgInfoList](#_查询组织信息列表)、[NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找)接口配套使用
* 此接口需多次调用，逐个获取组织信息；
* 获取组织信息结束后必须调用[NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找)接口，以释放资源,关闭查找；

**See Also：**

[NETDEV\_FindOrgInfoList](#_查询组织信息列表)、[NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找)

#### 关闭组织信息查找

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseOrgInfo  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭组织信息查找,释放资源

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindOrgInfoList](#_查询组织信息列表)、[NETDEV\_FindNextOrgInfo](#_逐个获取查找到的组织信息)接口配套使用
* 获取组织信息结束后必须调用[NETDEV\_FindCloseOrgInfo](#_关闭组织信息查找)接口，以释放资源,关闭查找；

**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)

## 网络配置

### 获取NAT类型

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetNATType  (  CHAR\* pszDomain,  INT32 \*pdwNatType  ); |

**接口描述：**

获取NAT类型

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| pszDomain | IN | 域名 |
| pdwNatType | OUT | NAT类型 参考#[NETDEV\_NAT\_TYPE\_E](#_NAT类型枚举) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

### 获取映射端口

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetUpnpNatState  (  LPVOID lpUserID,  [LPNETDEV\_UPNP\_NAT\_STATE\_S](#_网络端口号状态信息结构体) pstNatState  ); |

**接口描述：**

获取映射端口

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstNatState | OUT | 网络端口号状态信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SetUpnpNatState](#_设置映射端口)

### 设置映射端口

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetUpnpNatState  (  LPVOID lpUserID,  [LPNETDEV\_UPNP\_NAT\_STATE\_S](#_网络端口号状态信息结构体) pstNatState  ); |

**接口描述：**

设置映射端口

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄 |
| pstNatState | IN | 网络端口号状态信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_GetUpnpNatState](#_获取映射端口)

## 解码设备

### 获取能力集

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetCapability  (  LPVOID lpUserID,  [LPNETDEV\_XW\_CAP\_INFO\_S](#_XW设备能力集信息结构体) pstCapInfo  ); |

**接口描述：**

获取能力集；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstCapInfo | OUT | 电视墙信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取本地编码通道数量

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetChannelsNum  (  LPVOID lpUserID,  [LPNETDEV\_XW\_CHANNELS\_NUM\_S](#_通道数量信息结构体) pstChannels  ); |

**接口描述：**

获取本地编码通道数量；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstChannels | OUT | 通道信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取视频通道信息列表

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetChannelsInfoList  (  LPVOID lpUserID,  [LPNETDEV\_XW\_CHANNELS\_LIST\_S](#_解码器通道信息结构体) pstChannelsList  ); |

**接口描述：**

获取视频通道信息列表；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstChannelsList | INOUT | 编码通道信息，通道数量为输入信息，通道信息为输出信息，内存由调用者分配 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 通过[NETDEV\_XW\_GetChannelsNum](#_获取本地编码通道数量)接口获取通道数量；

**See Also：**

[NETDEV\_XW\_GetChannelsNum](#_获取本地编码通道数量)

### 创建电视墙配置

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_CreateTVWallCfg  (  LPVOID lpUserID,  [LPNETDEV\_XW\_TVWALL\_CFG\_S](#_电视墙信息结构体) pstTVWallCfg,  UINT32 \*pudwLastChange  ); |

**接口描述：**

创建电视墙配置；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstTVWallCfg | INOUT | 电视墙信息，设备端创建成功后返回电视墙ID、资源序号OrderNo |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 修改电视墙配置

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_ModifyTVWallCfg  (  LPVOID lpUserID,  [LPNETDEV\_XW\_TVWALL\_CFG\_S](#_电视墙信息结构体) pstTVWallCfg,  UINT32 \*pudwLastChange  ); |

**接口描述：**

修改电视墙配置；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstTVWallCfg | IN | 电视墙信息，结构体中需要输入电视墙ID |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 删除电视墙配置

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteTVWallCfg  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 \*pudwLastChange  ); |

**接口描述：**

删除电视墙配置；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取电视墙配置

#### 获取电视墙配置列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindTVWallCfgList  (  LPVOID lpUserID  ); |

**接口描述：**

获取电视墙配置列表；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_XW\_FindNextTVWallCfg](#_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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_XW\_FindNextTVWallCfg](#_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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_XW\_FindNextTVWallCfg](#_逐个获取查找到的电视墙配置信息)接口获取设备电视墙配置信息；
* 获取电视墙配置信息结束后必须调用[NETDEV\_XW\_FindCloseTVWallCfg](#_关闭查找电视墙配置信息)接口，以释放资源；

**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)

#### 逐个获取查找到的电视墙配置信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextTVWallCfg  (  LPVOID lpFindHandle,  [LPNETDEV\_XW\_TVWALL\_CFG\_S](#_电视墙信息结构体) pstTVWallCfg  ); |

**接口描述：**

逐个获取查找到的电视墙配置信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindTVWallCfgList](#_获取电视墙配置列表)接口的返回值 |
| pstTVWallCfg | OUT | 电视墙配置信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口需循环调用获取设备电视墙配置信息；
* 获取电视墙配置信息结束后必须调用[NETDEV\_XW\_FindCloseTVWallCfg](#_关闭查找电视墙配置信息)接口，以释放资源；

**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)

#### 关闭查找电视墙配置信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseTVWallCfg  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭查找电视墙配置信息，释放资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindTVWallCfgList](#_获取电视墙配置列表)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 获取电视墙配置信息结束后必须调用[NETDEV\_XW\_FindCloseTVWallCfg](#_关闭查找电视墙配置信息)接口，以释放资源；

**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)

### 获取单个电视墙布局配置信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetTVWallCfg  (  LPVOID lpUserID,  UINT32 udwTVWallID,  [LPNETDEV\_XW\_TVWALL\_CFG\_S](#_电视墙信息结构体) pstTVWallCfg  ); |

**接口描述：**

获取单个电视墙布局配置信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTVWallID | IN | 电视墙ID |
| pstTVWallCfg | INOUT | 电视墙信息，若存在电视墙编码字段，则需输入szTVWallCode |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 创建单个虚拟LED

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

创建单个虚拟LED；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstVirtualLEDInfo | INOUT | 虚拟LED信息,LEDID由设备返回 |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 删除单个虚拟LED

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteOneVirtualLED  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwLedlID,  UINT32 \*pudwLastChange  ); |

**接口描述：**

删除单个虚拟LED；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| udwLedlID | IN | 虚拟LED的ID |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 配置单个虚拟LED

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

配置单个虚拟LED；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstVirtualLEDInfo | IN | 虚拟LED信息 |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取单个虚拟LED

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetVirtualLED  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_VIRTUAL\_LED\_INFO\_S](#_虚拟LED信息结构体) pstVirtualLEDInfo  ); |

**接口描述：**

获取单个虚拟LED；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstVirtualLEDInfo | INOUT | 虚拟LED信息,需要输入LED ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取虚拟LED列表信息

#### 获取虚拟LED列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindVirtualLEDList  (  LPVOID lpUserID,  UINT32 udwTvWallID  ); |

**接口描述：**

获取虚拟LED列表；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_XW\_FindNextVirtualLED](#_逐个获取虚拟LED信息)、[NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_XW\_FindNextVirtualLED](#_逐个获取虚拟LED信息)、[NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_XW\_FindNextVirtualLED](#_逐个获取虚拟LED信息)接口获取虚拟LED信息；
* 获取虚拟LED信息结束后必须调用[NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息)接口，以释放资源；

**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)

#### 逐个获取虚拟LED信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextVirtualLED  (  LPVOID lpFindHandle,  [LPNETDEV\_XW\_VIRTUAL\_LED\_INFO\_S](#_虚拟LED信息结构体) pstVirtualLEDInfo  ); |

**接口描述：**

逐个获取虚拟LED；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindVirtualLEDList](#_获取虚拟LED列表)接口的返回值 |
| pstVirtualLEDInfo | OUT | 虚拟LED信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_XW\_FindVirtualLEDList](#_获取虚拟LED列表)、[NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息)接口配套使用；
* 该接口需循环调用获取虚拟LED信息；
* 获取虚拟LED信息结束后必须调用[NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息)接口，以释放资源；

**See Also：**

[NETDEV\_XW\_FindVirtualLEDList](#_获取虚拟LED列表)、[NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息)

#### 关闭查找虚拟LED信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseVirtualLED  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭虚拟LED列表；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindVirtualLEDList](#_获取虚拟LED列表)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_XW\_FindVirtualLEDList](#_获取虚拟LED列表)、[NETDEV\_XW\_FindNextVirtualLED](#_逐个获取虚拟LED信息)接口配套使用；
* 获取虚拟LED信息结束后必须调用[NETDEV\_XW\_FindCloseVirtualLED](#_关闭查找虚拟LED信息)接口，以释放资源；

**See Also：**

[NETDEV\_XW\_FindVirtualLEDList](#_获取虚拟LED列表)、[NETDEV\_XW\_FindNextVirtualLED](#_逐个获取虚拟LED信息)

### 创建单个窗口配置信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

创建单个窗口配置信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstWndInfo | INOUT | 窗口信息,窗口ID由设备端分配并输出 |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 批量开窗

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量开窗；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwTimeOut | IN | 超时时间(秒)，范围10s~60s |
| udwTVWallID | IN | 电视墙ID |
| pstWndInfo | IN | 窗口信息结构体，创建成功后返回窗口ID |
| pstResultWndInfo | OUT | 开窗返回信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 修改单个窗口配置信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

修改单个窗口配置信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstWndInfo | INOUT | 窗口信息,需要输入窗口ID |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 批量修改窗口

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量修改窗口；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTVWallID | IN | 电视墙ID |
| pstWndInfo | IN | 窗口信息 |
| pstResultWndInfo | OUT | 修改窗口返回信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 删除单个窗口配置信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteWnd  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwWndID,  UINT32 \*pudwLastChange  ); |

**接口描述：**

删除单个窗口配置信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| udwWndID | IN | 窗口ID |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 电视墙窗口放大/还原

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

电视墙窗口放大/还原；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| udwWndID | IN | 窗口ID |
| pstScreenZoomInfo | IN | 屏幕放大信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 批量关窗

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量关窗；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwTimeOut | IN | 超时时间(秒)，范围10s~60s |
| udwTVWallID | IN | 电视墙ID |
| pstDeleteWndInfo | INOUT | 窗口信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取窗口配置列表

#### 获取场景下所有窗口配置信息

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindWndList  (  LPVOID lpUserID,  UINT32 udwTvWallID  ); |

**接口描述：**

获取场景下所有窗口配置信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_XW\_FindNextWnd](#_逐个获取查找到的窗口配置信息)接口获取窗口信息；
* 获取窗口信息结束后必须调用[NETDEV\_XW\_FindCloseWnd](#_关闭查找窗口配置)接口，以释放资源；

**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)

#### 逐个获取查找到的窗口配置信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextWnd  (  LPVOID lpFindHandle,  [LPNETDEV\_XW\_SCENE\_WND\_INFO\_S](#_窗口信息结构体) pstWndInfo  ); |

**接口描述：**

逐个获取查找到的窗口配置信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindWndList](#_获取场景下所有窗口配置信息)接口的返回值 |
| pstWndInfo | OUT | 窗口配置信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口需循环调用获取窗口信息；
* 获取窗口信息结束后必须调用[NETDEV\_XW\_FindCloseWnd](#_关闭查找窗口配置)接口，以释放资源；

**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)

#### 关闭查找窗口配置

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseWnd  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭查找窗口配置，释放资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindWndList](#_获取场景下所有窗口配置信息)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 获取窗口信息结束后必须调用[NETDEV\_XW\_FindCloseWnd](#_关闭查找窗口配置)接口，以释放资源；

**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)

### 获取单个窗口配置信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetWnd  (  LPVOID lpUserID,  UINT32 udwTvWallID,  [LPNETDEV\_XW\_SCENE\_WND\_INFO\_S](#_窗口信息结构体) pstWndInfo  ); |

**接口描述：**

获取单个窗口配置信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstWndInfo | INOUT | 窗口信息,需要输入窗口ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取单个实况业务绑定信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取单个实况业务绑定信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstDisplayerID | IN | 电视墙ID、窗口ID、分屏序号 |
| pstVideoSource | OUT | 摄像机业务信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 绑定单个实况业务

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

绑定单个实况业务；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstDisplayerID | IN | 电视墙ID、窗口ID、分屏序号 |
| pstVideoSource | IN | 摄像机业务信息 |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 删除单个实况业务绑定

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

删除单个实况业务绑定；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstDisplayerID | IN | 电视墙ID、窗口ID、分屏序号 |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取场景下的实况业务绑定信息

#### 获取场景下的实况业务绑定信息

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindLiveBindList  (  LPVOID lpUserID,  UINT32 udwTvWallID  ); |

**接口描述：**

获取场景下的实况业务绑定信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_XW\_FindNextLiveBindInfo](#_逐个获取实况业务绑定信息)接口获取实况业务绑定信息；
* 获取实况业务绑定信息结束后必须调用[NETDEV\_XW\_FindCloseLiveBindInfo](#_关闭获取实况业务绑定信息)接口，以释放资源；

**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)

#### 逐个获取实况业务绑定信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextLiveBindInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_XW\_VIDEO\_SOURCE\_S](#_视频源信息结构体) pstLiveBindSource  ); |

**接口描述：**

逐个获取实况业务绑定信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindLiveBindList](#_获取场景下的实况业务绑定信息)接口的返回值 |
| pstLiveBindSource | OUT | 业务流信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口需循环调用获取实况业务绑定信息；
* 获取实况业务绑定信息结束后必须调用[NETDEV\_XW\_FindCloseLiveBindInfo](#_关闭获取实况业务绑定信息)接口，以释放资源；

**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)

#### 关闭获取实况业务绑定信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseLiveBindInfo  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭实况业务绑定信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindLiveBindList](#_获取场景下的实况业务绑定信息)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 获取实况业务绑定信息结束后必须调用[NETDEV\_XW\_FindCloseLiveBindInfo](#_关闭获取实况业务绑定信息)接口，以释放资源；

**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)

### 开始被动解码

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

开始被动解码；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstDisplayerID | IN | 设备ID信息 |
| pstVideoSource | IN | 视频源信息 |
| pudwTaskNo | OUT | 业务号，用于控制被动解码 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_StopPassiveDecode](#_停止被动解码)、[NETDEV\_SendPassiveData](#_向被动解码通道发送数据)

### 向被动解码通道发送数据

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

向被动解码通道发送数据；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTaskNo | IN | 业务号，用于控制被动解码 |
| pstSendData | IN | 待发送数据内容 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_StartPassiveDecode](#_开始被动解码)、[NETDEV\_StopPassiveDecode](#_停止被动解码)

### 停止被动解码

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopPassiveDecode  (  LPVOID lpUserID,  UINT32 udwTaskNO  ); |

**接口描述：**

停止被动解码；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTaskNO | IN | 业务号，用于控制被动解码 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_StartPassiveDecode](#_开始被动解码)、[NETDEV\_SendPassiveData](#_向被动解码通道发送数据)

### 业务流信息

#### 获取所有业务流信息

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindStreamList  (  LPVOID lpUserID,  UINT32 udwTvWallID  ); |

**接口描述：**

获取所有业务流信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_XW\_FindNextStreamInfo](#_逐个获取业务流信息)、[NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_XW\_FindNextStreamInfo](#_逐个获取业务流信息)、[NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_XW\_FindNextStreamInfo](#_逐个获取业务流信息)接口获取业务流信息；
* 获取业务流信息结束后必须调用[NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息)接口，以释放资源；

**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)

#### 逐个获取业务流信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextStreamInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_XW\_STREAM\_INFO\_S](#_流业务的详细信息结构体) pstXWStreamInfo  ); |

**接口描述：**

逐个业务流信息获取；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindStreamList](#_获取所有业务流信息)接口的返回值 |
| pstXWStreamInfo | OUT | 业务流信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_XW\_FindStreamList](#_获取所有业务流信息)、[NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息)接口配套使用；
* 该接口需循环调用获取业务流信息；
* 获取业务流信息结束后必须调用[NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息)接口，以释放资源；

**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)

#### 关闭获取业务流信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseStreamInfo  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭获取业务流信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindStreamList](#_获取所有业务流信息)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_XW\_FindStreamList](#_获取所有业务流信息)、[NETDEV\_XW\_FindNextStreamInfo](#_逐个获取业务流信息)接口配套使用；
* 获取业务流信息结束后必须调用[NETDEV\_XW\_FindCloseStreamInfo](#_关闭获取业务流信息)接口，以释放资源；

**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)

### 获取设备全部轮巡资源

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取设备全部轮巡资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwResNum | INOUT | 轮巡资源个数，作为入参时，用来指定pstSequenceResInfoList数组的大小 |
| pstSequenceResInfoList | OUT | 轮巡资源基本信息列表指针，内存由用户维护 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* udwResNum返回实际获取到的轮巡资源个数，如果接口调用失败并且错误码为NETDEV\_E\_NEEDMOREDATA时，说明用户分配的内存不够；

### 添加单个轮巡资源信息到设备

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

添加单个轮巡资源信息到设备；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstSeqResInfo | INOUT | 轮巡资源详细信息 |
| pudwLastChange | OUT | 摘要字 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 修改单个轮巡资源详细信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

修改单个轮巡资源详细信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstSeqResInfo | IN | 轮巡资源详细信息 |
| pudwLastChange | OUT | 摘要字 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 删除单个轮巡资源信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteSequenceResource  (  LPVOID lpUserID,  UINT32 udwSeqResID,  UINT32\* pudwLastChange  ); |

**接口描述：**

删除单个轮巡资源信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwSeqResID | IN | 轮巡资源ID |
| pudwLastChange | OUT | 摘要字 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取单个轮巡资源详细信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取单个轮巡资源详细信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstSeqResInfo | INOUT | 轮巡资源详细信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 创建单个场景信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

创建单个场景信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstSceneBase | INOUT | 输入场景名称，输出场景ID 、资源序号OrderNo |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 修改场景信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

修改场景信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstSceneModifyInfo | INOUT | 修改信息 |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 删除场景信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteSceneInfo  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwSceneID,  UINT32 \*pudwLastChange  ); |

**接口描述：**

删除场景信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| udwSceneID | IN | 场景ID |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取当前场景的关联场景

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_GetCurrentSceneID  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 \*pudwSceneID  ); |

**接口描述：**

获取当前场景的关联场景；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pudwSceneID | OUT | 场景ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 切换场景

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_SetCurrentSceneID  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwSceneID,  UINT32 \*pudwLastChange  ); |

**接口描述：**

切换为当前场景；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| udwSceneID | IN | 场景ID |
| pudwLastChange | OUT | 摘要字，配置改动会更新摘要字，由设备端生成 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取当前场景配置内容

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取当前场景配置内容；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstSceneDetailInfo | INOUT | 场景信息,若存在电视墙编码字段，则需要输入szTVWallCode |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取场景信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取场景信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwSceneID | IN | 场景ID |
| pstSceneDetailInfo | INOUT | 场景详细信息,若存在电视墙编码字段，则输入szTVWallCode |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取场景配置列表

#### 获取场景配置列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindSceneCfgList  (  LPVOID lpUserID,  UINT32 udwTvWallID  ); |

**接口描述：**

获取场景配置列表；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_XW\_FindNextSceneCfg](#_逐个获取查找到的场景配置信息)接口获取场景信息；
* 获取场景信息结束后必须调用[NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景)接口，以释放资源；

**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)

#### 逐个获取查找到的场景配置信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindNextSceneCfg  (  LPVOID lpFindHandle,  LPNETDEV\_XW\_SCENE\_INFO\_BASIC\_S pstSceneShortInfo  ); |

**接口描述：**

逐个获取查找到的场景配置信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindSceneCfgList](#_获取场景配置列表)接口的返回值 |
| pstSceneShortInfo | OUT | 场景信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_XW\_FindSceneCfgList](#_获取场景配置列表)、[NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景)接口配套使用；
* 该接口需循环调用获取场景信息；
* 获取场景信息结束后必须调用[NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景)接口，以释放资源；

**See Also：**

[NETDEV\_XW\_FindSceneCfgList](#_获取场景配置列表)、[NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景)

#### 关闭查找场景

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseSceneCfg  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭查找场景，释放资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindSceneCfgList](#_获取场景配置列表)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 获取场景信息结束后必须调用[NETDEV\_XW\_FindCloseSceneCfg](#_关闭查找场景)接口，以释放资源；

**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)

### 绑定轮巡资源到当前场景布局

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

绑定轮巡资源到当前场景布局；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTVWallID | IN | 电视墙ID |
| pstSeqResBindInfo | IN | 轮巡资源绑定信息 |
| pudwLastChange | OUT | 摘要字 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 修改当前场景轮巡布局

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

修改当前场景轮巡布局；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTVWallID | IN | 电视墙ID |
| pstSeqResBindInfo | IN | 轮巡资源绑定信息 |
| pudwLastChange | OUT | 摘要字 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 删除当前场景轮巡布局

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteSequenceResToScene  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwSeqResBindID,  UINT32\* pudwLastChange  ); |

**接口描述：**

删除当前场景轮巡布局；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| udwSeqResBindID | IN | 轮巡资源绑定信息ID |
| pudwLastChange | OUT | 摘要字 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 轮巡播放控制

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

轮巡播放控制；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstSeqPlayCtrlInfo | IN | 轮巡播放控制信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 添加单个场景轮巡计划

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

添加单个场景轮巡计划；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstScenesPlanInfo | INOUT | 场景计划列表信息指针，输出资源ID、资源序号OrderNo |
| pudwLastChange | OUT | 摘要字 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 修改单个场景轮巡计划

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

修改单个场景轮巡计划；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstScenesPlanInfo | IN | 场景计划列表信息 |
| pudwLastChange | OUT | 摘要字 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 删除单个场景轮巡计划

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_DeleteSencesSequencePlan  (  LPVOID lpUserID,  UINT32 udwTvWallID,  UINT32 udwScenesSeqPlanID,  UINT32\* pudwLastChange  ); |

**接口描述：**

删除单个场景轮巡计划；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| udwScenesSeqPlanID | IN | 场景轮巡计划ID |
| pudwLastChange | OUT | 摘要字 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取单个场景轮巡计划

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取单个场景轮巡计划；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstScenesPlanInfo | INOUT | 场景计划列表信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取场景轮巡计划列表

#### 获取场景轮巡计划列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_XW\_FindSencesSequencePlanList  (  LPVOID lpUserID,  UINT32 udwTvWallID  ); |

**接口描述：**

获取场景轮巡计划列表；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_XW\_FindNextSencesSequencePlan](#_逐个获取场景轮巡计划)接口获取场景轮巡计划信息；
* 获取场景轮巡计划信息结束后必须调用[NETDEV\_XW\_FindCloseSencesSequencePlan](#_释放获取场景轮巡计划时内存资源)接口，以释放资源；

**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)

#### 逐个获取场景轮巡计划

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

逐个获取场景轮巡计划；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindSencesSequencePlanList](#_获取场景轮巡计划列表)接口的返回值 |
| pstScenesPlanInfo | INOUT | 场景计划信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口需循环调用获取场景轮巡计划信息；
* 获取场景轮巡计划信息结束后必须调用[NETDEV\_XW\_FindCloseSencesSequencePlan](#_释放获取场景轮巡计划时内存资源)接口，以释放资源；

**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)

#### 释放获取场景轮巡计划时内存资源

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_XW\_FindCloseSencesSequencePlan  (  LPVOID lpFindHandle  ); |

**接口描述：**

释放获取场景轮巡计划时内存资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_XW\_FindSencesSequencePlanList](#_获取场景轮巡计划列表)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 获取场景轮巡计划信息结束后必须调用[NETDEV\_XW\_FindCloseSencesSequencePlan](#_释放获取场景轮巡计划时内存资源)接口，以释放资源；

**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)

### 场景轮巡计划播放控制

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

场景轮巡计划播放控制；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTvWallID | IN | 电视墙ID |
| pstScenesPlanPlayCtrlInfo | IN | 轮巡播放控制信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

## 智能功能

### 获取系统图片信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetSystemPicture  (  LPVOID lpUserID,  CHAR\* pszURL,  UINT32 udwSize,  CHAR\* pszdata  ); |

**接口描述：**

获取系统图片信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pszURL | IN | 图片URL |
| udwSize | IN | 加密前数据大小 |
| pszdata | OUT | 图片数据，需动态分配内存 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 手动联动操作

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ManualLink  (  LPVOID lpUserID,  UINT32 udwChannelID,  [LPNETDEV\_MANUAL\_LINK\_S](#_手动联动数据结构体) pstManualLink  ); |

**接口描述：**

手动联动操作；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwChannelID | IN | 通道号 |
| pstManualLink | IN | 手动联动数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 注册车牌识别上报回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetCarPlateCallBack  (  LPVOID lpUserID,  NETDEV\_CarPlateCallBack\_PF cbCarPlateCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册车牌识别上报回调函数；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| cbCarPlateCallBack | IN | 需要注册的回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstCarPlateData | IN | 车牌识别数据 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 注册人脸识别报警消息回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetFaceAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_FaceAlarmMessCallBack\_PF cbFaceAlarmMessCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册人脸识别报警消息回调函数；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| cbFaceAlarmMessCallBack | IN | 回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstFaceAlarmReportInfo | IN | 上报信息 |
| lpBuf | IN | 缓冲区指针 |
| dwBufLen | IN | 告警信息对应结构体的长度 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 注册人脸抓拍上报回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetFaceSnapshotCallBack  (  LPVOID lpUserID,  NETDEV\_FaceSnapshotCallBack\_PF cbFaceSnapshotCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册人脸抓拍上报回调函数；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| cbFaceSnapshotCallBack | IN | 需要注册的回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstFaceSnapShotData | IN | 人脸抓拍数据 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 注册热度图上报回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetHeatMapCallBack  (  LPVOID lpUserID,  NETDEV\_HeatMapCallBack\_PF cbHeatMapCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册热度图上报回调函数；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| cbHeatMapCallBack | IN | 需要注册的回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstHeatMapData | IN | 热度图数据 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 注册人脸报警消息回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPersonAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_PersonAlarmMessCallBack\_PF cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册人脸报警消息回调函数；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| cbAlarmMessCallBack | IN | 回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstAlarmData | IN | 告警数据 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 注册结构化报警消息回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetStructAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_StructAlarmMessCallBack\_PF cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册结构化报警消息回调函数；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| cbAlarmMessCallBack | IN | 回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstAlarmInfo | IN | 告警信息 |
| pstAlarmData | IN | 告警数据 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 注册车牌识别报警回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetVehicleAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_VehicleAlarmMessCallBack\_PF cbVehicleAlarmMessCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册回调函数，接收车牌识别报警消息等；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| cbVehicleAlarmMessCallBack | IN | 回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstVehicleAlarmInfo | IN | 上报信息 |
| lpBuf | IN | 缓冲区指针 |
| dwBufLen | IN | 告警信息对应结构体的长度 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 订阅智能事件

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

订阅智能事件；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstSubscribeInfo | IN | 订阅信息 |
| pstSmartInfo | INOUT | 智能事件信息，成功返回订阅ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 订阅前需要先调用[NETDEV\_SetAlarmCallBack\_V30](#_注册回调函数,接收报警消息等)接口注册回调函数；

**See Also：**

[NETDEV\_UnsubscribeSmart](#_取消订阅智能事件)

### 取消订阅智能事件

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_UnsubscribeSmart  (  LPVOID lpUserID,  [LPNETDEV\_SMART\_INFO\_S](#_智能事件信息结构体) pstSmartInfo  ); |

**接口描述：**

取消订阅智能事件；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstSmartInfo | IN | 智能事件 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SubscribeSmart](#_订阅智能事件)

### LAPI告警订阅

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

LAPI告警订阅；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstSubInfo | IN | 告警订阅请求 |
| pstSubSuccInfo | OUT | 订阅成功返回信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_UnSubLapiAlarm](#_取消LAPI告警订阅)

### 取消LAPI告警订阅

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_UnSubLapiAlarm  (  LPVOID lpUserID,  UINT32 udwID  ); |

**接口描述：**

取消LAPI告警订阅；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwID | IN | 告警订阅ID，[NETDEV\_SubscibeLapiAlarm](#_LAPI告警订阅)接口中获取 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**See Also：**

[NETDEV\_SubscibeLapiAlarm](#_LAPI告警订阅)

### 人脸识别记录

#### 查询人脸识别记录

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

查询人脸识别记录；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstFindCond | IN | 查询条件 |
| pstResultInfo | OUT | 人脸识别记录信息 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextFaceRecordDetail](#_逐个获取人脸识别记录)、[[NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录)](#_关闭查找人脸识别记录)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextFaceRecordDetail](#_逐个获取人脸识别记录)、[NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextFaceRecordDetail](#_逐个获取人脸识别记录)接口获取日志信息；
* 获取日志信息结束后必须调用[NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录)接口，以释放资源,关闭查找；

**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)

#### 逐个获取人脸识别记录

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextFaceRecordDetail  (  LPVOID lpFindHandle,  [LPNETDEV\_FACE\_RECORD\_SNAPSHOT\_INFO\_S](#_人脸识别记录结构体) pstRecordInfo  ); |

**接口描述：**

逐个获取查找到的人脸识别记录信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录)接口返回值 |
| pstRecordInfo | OUT | 保存人脸识别记录信息的指针 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录)、[NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录)接口配套使用；
* 此接口需循环调用获取日志信息；
* 获取日志信息结束后必须调用[NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录)接口，以释放资源,关闭查找；

**See Also：**

[NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录)、[NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录)

#### 关闭查找人脸识别记录

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseFaceRecordDetail  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭查找人脸识别记录，释放资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录)接口返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录)、[NETDEV\_FindNextFaceRecordDetail](#_逐个获取人脸识别记录)接口配套使用；
* 获取日志信息结束后必须调用[NETDEV\_FindCloseFaceRecordDetail](#_关闭查找人脸识别记录)接口，以释放资源,关闭查找；

**See Also：**

[NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录)、[NETDEV\_FindNextFaceRecordDetail](#_逐个获取人脸识别记录)

#### 查询单个人脸识别记录的人脸图片信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetFaceRecordImageInfo  (  LPVOID lpUserID,  UINT32 udwRecordID,  UINT32 udwFaceImageType,  [LPNETDEV\_FILE\_INFO\_S](#_文件信息结构体) pstFileInfo  ); |

**接口描述：**

查询单个人脸识别记录的人脸图片信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwRecordID | IN | 人脸识别告警记录ID |
| udwFaceImageType | IN | 人脸通行记录类型 |
| pstFileInfo | INOUT | 人脸图片信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 停车场

#### 获取某停车场下所有出入口信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取某停车场下所有出入口信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwParkID | IN | 停车场ID |
| pstParkEntranceInfoList | OUT | 出入口信息列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 批量获取停车场信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetParkInfoList  (  LPVOID lpUserID,  [LPNETDEV\_PARK\_INFO\_LIST\_S](#_停车场信息列表结构体) pstParkInfoList  ); |

**接口描述：**

批量获取停车场信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstParkInfoList | OUT | 停车场信息列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 获取停车场出入口所有车道信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取停车场出入口所有车道信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstParkEntranceID | IN | 停车场出入口ID |
| pstVehicleLaneList | OUT | 出入口信息列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 下发第三方支付订单

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

下发第三方支付订单；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstPayOrder | IN | 订单信息 |
| pstCost | OUT | 停车费用信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 注册停车场事件回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetParkEventCallBack  (  LPVOID lpUserID,  NETDEV\_ParkEventMessCallBack\_PF cbParkAlarmMessCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册回调函数，接收停车场事件信息等；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| cbParkAlarmMessCallBack | IN | 接收数据回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| stParkAlarmInfo | IN | 上报信息 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 时间模板

#### 添加时间模板

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddTimeTemplate  (  LPVOID lpUserID,  [LPNETDEV\_SYSTEM\_TIME\_TEMPLATE\_S](#_时间模板配置结构体) pstTimeTemplate,  UINT32 \*pudwTemplateID  ); |

**接口描述：**

添加时间模板；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstTimeTemplate | IN | 时间模板信息 |
| pudwTemplateID | OUT | 时间模板ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* VMS不支持指定模板ID；
* 如果传入的模板ID不为0; 则直接使用模板ID当索引,并返回，如果传入的模板ID是已经存在的值，则接口返回失败；
* 如果传入的模板ID 为0; 则终端分配一个唯一的ID,并返回；

#### 批量删除指定的时间模板

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量删除指定的时间模板；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstTemplateIDList | IN | 删除的时间模板ID列表 |
| pstDeleteResults | OUT | 批量删除的结果信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 批量删除仅VMS支持；
* pstDeleteResults->udwNum不应小于pstTemplateIDList->udwNum；
* 出参返回：udwNum、udwID、udwResultCode；

#### 批量获取时间模板信息

##### 批量获取指定模板类型的时间模板信息

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindTimeTemplateByTypeList  (  LPVOID lpUserID,  UINT32 udwTemplateType  ); |

**接口描述：**

批量获取指定模板类型的时间模板信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwTemplateType | IN | 时间模板类型 仅VMS支持 0:录像时间模板 1:报警时间模板 2:用户时间模板 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextTimeTemplateByTypeInfo](#_逐个获取按类型找到的时间模板信息)接口获取日志信息；
* 获取日志信息结束后必须调用[NETDEV\_FindCloseTimeTemplateByTypeList](#_关闭查找按类型找到的时间模板列表)接口，以释放资源,关闭查找；

**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)

##### 逐个获取按类型找到的时间模板信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextTimeTemplateByTypeInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_TIME\_TEMPLATE\_BASE\_INFO\_S](#_时间模板结构体) pstTimeTemplate  ); |

**接口描述：**

逐个获取按类型找到的时间模板信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_FindTimeTemplateByTypeList](#_批量获取指定模板类型的时间模板信息)接口返回值 |
| pstTimeTemplate | OUT | 时间模板 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 此接口需循环调用获取日志信息；
* 获取日志信息结束后必须调用[NETDEV\_FindCloseTimeTemplateByTypeList](#_关闭查找按类型找到的时间模板列表)接口，以释放资源,关闭查找；

**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)

##### 关闭查找按类型找到的时间模板列表

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseTimeTemplateByTypeList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭查找按类型找到的时间模板列表，释放资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_FindTimeTemplateByTypeList](#_批量获取指定模板类型的时间模板信息)接口返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 获取日志信息结束后必须调用[NETDEV\_FindCloseTimeTemplateByTypeList](#_关闭查找按类型找到的时间模板列表)接口，以释放资源,关闭查找；

**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)

#### 获取指定时间模板信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetTimeTemplate  (  LPVOID lpUserID,  [LPNETDEV\_SYSTEM\_TIME\_TEMPLATE\_S](#_时间模板配置结构体) pstTimeTemplate  ); |

**接口描述：**

获取指定时间模板信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstTimeTemplate | INOUT | 时间模板信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 修改指定时间模板信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyTimeTemplate  (  LPVOID lpUserID,  [LPNETDEV\_SYSTEM\_TIME\_TEMPLATE\_S](#_时间模板配置结构体) pstTimeTemplate  ); |

**接口描述：**

修改指定事件模板信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstTimeTemplate | IN | 时间模板信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 门禁

#### 门禁人员管理

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ACSPersonCtrl  (  LPVOID lpUserID,  INT32 dwCommand,  [LPNETDEV\_ACS\_PERSON\_INFO\_S](#_门禁人员信息结构体) pstACSPersonInfo  ); |

**接口描述：**

门禁人员管理；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwCommand | IN | 门禁人员管理命令，参考[NETDEV\_ACS\_PERSON\_COMMOND\_TYPE\_E](#_门禁人员管理命令枚举) |
| pstACSPersonInfo | INOUT | 门禁人员信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 访客签离

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ACSPersonSignout  (  LPVOID lpUserID,  UINT32 udwPersonID  ); |

**接口描述：**

访客签离；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwPersonID | IN | 人员编号 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 添加访客黑名单

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddACSPersonBlackList  (  LPVOID lpUserID,  [LPNETDEV\_ACS\_PERSON\_BLACKLIST\_INFO\_S](#_黑名单信息结构体) pstBlackListInfo,  UINT32 \*pUdwBlackListID  ); |

**接口描述：**

添加访客黑名单；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstBlackListInfo | IN | 黑名单信息 |
| pUdwBlackListID | OUT | 黑名单ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 绑定指定人员的门卡信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

绑定指定人员的门卡信息（即开卡）；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwACSPersonID | IN | 门禁人员信息编号 |
| pstACSPersonCardInfo | INOUT | 门禁卡信息，udwCardID为输出参数 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 批量添加人员信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量添加人员信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstACSPersonList | IN | 人员列表，其中单张图片大小为2M |
| pstResultList | OUT | 返回列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 添加人员授权组信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_AddACSPersonPermissionGroup  (  LPVOID lpUserID,  [LPNETDEV\_ACS\_PERMISSION\_INFO\_S](#_授权信息结构体) pstPermissionGroupInfo,  UINT32 \*pUdwGroupID  ); |

**接口描述：**

添加人员授权组信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstPermissionGroupInfo | IN | 授权组信息 |
| pUdwGroupID | OUT | 授权组ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 删除访客黑名单

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteACSPersonBlackList  (  LPVOID lpUserID,  [LPNETDEV\_OPERATE\_LIST\_S](#_批处理列表) pstBlackList  ); |

**接口描述：**

删除访客黑名单;

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstBlackList | OUT | 黑名单信息列表 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 删除指定人员的门卡信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteACSPersonCardInfo  (  LPVOID lpUserID,  UINT32 udwACSPersonID  ); |

**接口描述：**

删除指定人员的门卡信息，即退卡;

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwACSPersonID | IN | 门禁人员信息编号 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 批量删除门禁人员信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

批量删除门禁人员信息;

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstBatchCtrlInfo | INOUT | 批量控制信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 删除人员授权组信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

删除人员授权组信息;

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstPermissionIDList | IN | 权限ID数组 |
| pstResutList | OUT | 操作结果返回信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 门禁通道批量控制

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_DoorBatchCtrl  (  LPVOID lpUserID,  INT32 dwCommand,  [LPNETDEV\_OPERATE\_LIST\_S](#_批处理列表) pstBatchCtrlInfo  ); |

**接口描述：**

门禁通道批量控制；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwCommand | IN | 门禁通道控制命令，可参考[NETDEV\_DOORCTRL\_ACTION\_TYPE\_E](#_报警点通道控制命令枚举) |
| pstBatchCtrlInfo | INOUT | 批量控制信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 门禁通道控制

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_DoorCtrl  (  LPVOID lpUserID,  INT32 dwChannelID,  INT32 dwCommand  ); |

**接口描述：**

门禁通道控制；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwChannelID | IN | 通道号 |
| dwCommand | IN | 门禁通道控制命令，可参考[NETDEV\_DOORCTRL\_ACTION\_TYPE\_E](#_报警点通道控制命令枚举) |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 查询出入记录

##### 查询出入记录

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

查询出入记录；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstFindCond | IN | 出入记录查询条件 |
| pstResultInfo | OUT | 出入记录实际总条数 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextACSAttendanceLog](#_获取下一条出入记录)、

[NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextACSAttendanceLog](#_获取下一条出入记录)、[NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextACSAttendanceLog](#_获取下一条出入记录)接口获取出入记录信息；
* 获取出入记录结束后必须调用[NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源)接口，以释放资源,关闭查找；

**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)

##### 获取下一条出入记录

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextACSAttendanceLog  (  LPVOID lpFindHandle,  [LPNETDEV\_ACS\_ATTENDANCE\_LOG\_INFO\_S](#_出入记录信息结构体) pstACSLogInfo  ); |

**接口描述：**

获取下一条出入记录；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 出入记录信息列表句柄，[NETDEV\_FindACSAttendanceLogList](#_查询出入记录)接口返回值 |
| pstACSLogInfo | OUT | 出入记录信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindACSAttendanceLogList](#_查询出入记录)、[NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源)接口配套使用；
* 此接口需循环调用获取出入记录信息；
* 获取出入记录信息结束后必须调用[NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源)接口，以释放资源；

**See Also：**

[NETDEV\_FindACSAttendanceLogList](#_查询出入记录)、[NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源)

##### 关闭查询出入记录资源

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseACSAttendanceLogList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭查询出入记录资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 出入记录信息列表句柄，[NETDEV\_FindACSAttendanceLogList](#_查询出入记录)接口返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindACSAttendanceLogList](#_查询出入记录)、[NETDEV\_FindNextACSAttendanceLog](#_获取下一条出入记录)接口配套使用；
* 获取出入记录信息结束后必须调用[NETDEV\_FindCloseACSAttendanceLogList](#_关闭查询出入记录资源)接口，以释放资源

**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)

#### 查看门禁授组信息

##### 查看门禁授权组列表

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

查看门禁授权组列表;

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstQueryCond | IN | 查询条件 |
| pstResultInfo | OUT | 实际查询结果 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1)、

[NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1)、[NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1)接口获取门禁授权组信息；
* 获取门禁授权组信息结束后必须调用[NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源)接口，以释放资源；

**See Also：**

[NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1)、[NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源)

##### 获取下一条记录

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextACSPermissionGroupInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_ACS\_PERMISSION\_INFO\_S](#_授权信息结构体) pstACSPermissionInfo  ); |

**接口描述：**

获取下一条门禁授权组记录;

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 门禁授权组列表信息查找句柄，[NETDEV\_FindACSPermissionGroupList](#_查看门禁授权组列表)接口返回值 |
| pstACSPermissionInfo | OUT | 门禁授权组信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindACSPermissionGroupList](#_查看门禁授权组列表)、[NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源)接口配套使用；
* 此接口需循环调用获取门禁授权组信息；
* 获取门禁授权组信息结束后必须调用[NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源)接口，以释放资源；

**See Also：**

[NETDEV\_FindACSPermissionGroupList](#_查看门禁授权组列表)、[NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源)

##### 关闭查询记录资源

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseACSPermissionGroupList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭查询记录资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 门禁授权组列表信息查找句柄，[NETDEV\_FindACSPermissionGroupList](#_查看门禁授权组列表)接口返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindACSPermissionGroupList](#_查看门禁授权组列表)、[NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1)接口配套使用；
* 获取门禁授权组信息结束后必须调用[NETDEV\_FindCloseACSPermissionGroupList](#_关闭查询记录资源)接口，以释放资源；

**See Also：**

[NETDEV\_FindACSPermissionGroupList](#_查看门禁授权组列表)、[NETDEV\_FindNextACSPermissionGroupInfo](#_获取下一条记录_1)

#### 获取访客黑名单信息

##### 获取访客黑名单列表

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取访客黑名单列表;

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstQueryCond | IN | 查询条件 |
| pstResultInfo | OUT | 实际查询结果 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录)、[NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录)、[NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录)接口获取访客黑名单信息；
* 获取访客黑名单信息结束后必须调用[NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1)接口，以释放资源；

**See Also：**

[NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录)、[NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1)

##### 获取下一条记录

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextACSPersonBlackListInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_ACS\_PERSON\_BLACKLIST\_INFO\_S](#_黑名单信息结构体) pstBlackListInfo  ); |

**接口描述：**

获取下一条访客黑名单记录；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 访客黑名单列表信息查询句柄，[NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表)接口返回值 |
| pstBlackListInfo | OUT | 访客黑名单列表信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表)、[NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1)接口配套使用；
* 此接口需循环调用获取访客黑名单信息；
* 获取访客黑名单信息结束后必须调用[NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1)接口，以释放资源；

**See Also：**

[NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表)、[NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1)

##### 关闭查询记录资源

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseACSPersonBlackList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭查询记录资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 访客黑名单列表信息查询句柄，[NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表)接口返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表)、[NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录)接口配套使用；
* 获取访客黑名单信息结束后必须调用[NETDEV\_FindCloseACSPersonBlackList](#_关闭查询记录资源_1)接口，以释放资源；

**See Also：**

[NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表)、[NETDEV\_FindNextACSPersonBlackListInfo](#_获取下一条记录)

#### 获取门禁人员信息

##### 获取门禁人员信息列表

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取门禁人员信息列表；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstQueryCond | IN | 门禁人员查询条件 |
| pstResultInfo | OUT | 返回信息 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[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)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextACSPersonInfo](#_获取下一个门禁人员信息)接口获取门禁人员信息；
* 获取门禁人员信息结束后必须调用[NETDEV\_FindCloseACSPersonInfo](#_关闭门禁人员信息列表资源)接口，以释放资源,关闭查找；

**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)

##### 获取下一个门禁人员信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextACSPersonInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_ACS\_PERSON\_BASE\_INFO\_S](#_门禁人员基本信息结构体) pstACSPersonInfo  ); |

**接口描述：**

获取下一个门禁人员信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 门禁人员信息列表查询句柄，[NETDEV\_FindACSPersonList](#_获取门禁人员信息列表)接口返回值 |
| pstACSPersonInfo | OUT | 门禁人员信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 此接口需循环调用获取门禁人员信息；
* 获取门禁人员信息结束后必须调用[NETDEV\_FindCloseACSPersonInfo](#_关闭门禁人员信息列表资源)接口，以释放资源,关闭查找；

**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)

##### 关闭门禁人员信息列表资源

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseACSPersonInfo  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭门禁人员信息列表资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 门禁人员信息列表查询句柄，[NETDEV\_FindACSPersonList](#_获取门禁人员信息列表)接口返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[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)接口配套使用；
* 获取门禁人员信息结束后必须调用[NETDEV\_FindCloseACSPersonInfo](#_关闭门禁人员信息列表资源)接口，以释放资源,关闭查找；

**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)

#### 查询访客记录信息

##### 查询访客记录

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

查询访客记录；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstFindCond | IN | 访客记录查询条件 |
| pstResultInfo | OUT | 访客记录实际总条数 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录)、[NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录)、[NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录)接口获取访客记录信息；
* 获取访客记录信息结束后必须调用[NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源)接口，以释放资源；

**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)

##### 获取下一条访客记录

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextACSVisitLog  (  LPVOID lpFindHandle,  [LPNETDEV\_ACS\_VISIT\_LOG\_INFO\_S](#_访客记录信息结构体) pstACSLogInfo  ); |

**接口描述：**

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 出入记录信息列表查找句柄，[[NETDEV\_FindACSVisitLogList](#_查询访客记录)](#_查询访客记录)接口返回值 |
| pstACSLogInfo | OUT | 出入记录信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[[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)接口配套使用；
* 此接口需循环调用获取出入记录信息；
* 获取出入记录信息结束后必须调用[NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源)接口，以释放资源；

**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)

##### 关闭查询访客记录资源

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseACSVisitLog  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭查询访客记录资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 出入记录信息列表查找句柄，[NETDEV\_FindACSVisitLogList](#_查询访客记录)接口返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindACSVisitLogList](#_查询访客记录)、[NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录)接口配套使用；
* 获取出入记录信息结束后必须调用[NETDEV\_FindCloseACSVisitLog](#_关闭查询访客记录资源)接口，以释放资源；

**See Also：**

[NETDEV\_FindACSVisitLogList](#_查询访客记录)、[NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录)

#### 获取指定访客黑名单信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetACSPersonBlackList  (  LPVOID lpUserID,  [LPNETDEV\_ACS\_PERSON\_BLACKLIST\_INFO\_S](#_黑名单信息结构体) pstBlackListInfo  ); |

**接口描述：**

获取指定访客黑名单信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstBlackListInfo | INOUT | 黑名单信息，其中udwBlackListID作为入参传入 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 获取指定人员所持门禁卡信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取指定人员所持门禁卡信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwACSPersonID | IN | 门禁人员信息编号 |
| pstACSPersonCardInfo | OUT | 门禁卡信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 获取指定人员授权信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetACSPersonPermission  (  LPVOID lpUserID,  UINT32 udwPersonID,  [LPNETDEV\_ACS\_DOOR\_PERMISSION\_INFO\_S](#_门授权信息结构体) pstPermissionInfo  ); |

**接口描述：**

获取指定人员授权信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwPersonID | IN | 人员ID |
| pstPermissionInfo | OUT | 人员授权信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 获取单个授权组信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetSinglePermGroupInfo  (  LPVOID lpUserID,  UINT32 udwPermissionGroupID,  [LPNETDEV\_ACS\_PERMISSION\_INFO\_S](#_授权信息结构体) pstAcsPerssionInfo  ); |

**接口描述：**

获取单个授权组信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwPermissionGroupID | IN | 授权组id |
| pstAcsPerssionInfo | INOUT | 权限组信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 修改访客黑名单信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyACSPersonBlackList  (  LPVOID lpUserID,  [LPNETDEV\_ACS\_PERSON\_BLACKLIST\_INFO\_S](#_黑名单信息结构体) pstBlackListInfo  ); |

**接口描述：**

修改访客黑名单信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstBlackListInfo | IN | 黑名单信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 修改人员授权组信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyACSPersonPermissionGroup  (  LPVOID lpUserID,  [LPNETDEV\_ACS\_PERMISSION\_INFO\_S](#_授权信息结构体) pstPermissionInfo  ); |

**接口描述：**

修改人员授权组信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstPermissionInfo | IN | 授权组信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 设置指定人员授权信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetACSPersonPermission  (  LPVOID lpUserID,  UINT32 udwPersonID,  [LPNETDEV\_ACS\_DOOR\_PERMISSION\_INFO\_S](#_门授权信息结构体) pstPermissionInfo  ); |

**接口描述：**

设置指定人员授权信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwPersonID | IN | 人员ID |
| pstPermissionInfo | IN | 人员授权信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 注册速通门报警消息回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetAlarmFGCallBack  (  LPVOID lpUserID,  NETDEV\_AlarmMessFGCallBack\_PF cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册速通门报警消息回调函数;

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| cbAlarmMessCallBack | IN | 回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstAlarmData | IN | 告警信息 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 深度周界

#### 启动目标检索

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

启动目标检索；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstStartInfo | IN | 查询开始信息 |
| pudwSearchID | OUT | 业务号 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 获取目标检索查询进度

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_GetObjectSearchProg  (  LPVOID lpUserID,  UINT32 udwSearchID,  UINT32\* pudwPercent  ); |

**接口描述：**

获取目标检索进度；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwSearchID | IN | 业务号 |
| pudwPercent | OUT | 查询进度 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 查询目标检索结果

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

查询目标检索结果；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstQueryCond | IN | 查询条件 |
| pstQueryResult | OUT | 查询结果 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextObjectSearchInfo](#_逐个获取查找到的目标检索结果)、

[NETDEV\_FindCloseObjectSearchList](#_关闭目标检索结果查找,释放资源)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextObjectSearchInfo](#_逐个获取查找到的目标检索结果)、[NETDEV\_FindCloseObjectSearchList](#_关闭目标检索结果查找,释放资源)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextObjectSearchInfo](#_逐个获取查找到的目标检索结果)接口获取目标检索结果；
* 获取目标检索结果结束后必须调用[NETDEV\_FindCloseObjectSearchList](#_关闭目标检索结果查找,释放资源)接口，以释放资源；

#### 逐个获取查找到的目标检索结果

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextObjectSearchInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_OBJECT\_RESULT\_INFO\_S](#_目标结果信息) pstObjectResultInfo  ); |

**接口描述：**

查询目标检索结果；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_FindObjectSearchList](#_查询目标检索结果)接口的返回值 |
| pstObjectResultInfo | IN | 查询到的目标检索结果信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindObjectSearchList](#_查询目标检索结果)、[NETDEV\_FindCloseObjectSearchList](#_关闭目标检索结果查找,释放资源)接口配套使用；
* 该接口需循环调用获取目标检索结果；
* 获取目标检索结果结束后必须调用[NETDEV\_FindCloseObjectSearchList](#_关闭目标检索结果查找,释放资源)接口，以释放资源；

#### 关闭目标检索结果查找,释放资源

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseObjectSearchList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭目标检索结果查找,释放资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_FindObjectSearchList](#_查询目标检索结果)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindObjectSearchList](#_查询目标检索结果)、[NETDEV\_FindNextObjectSearchInfo](#_逐个获取查找到的目标检索结果)接口配套使用；
* 获取目标检索结果结束后必须调用[NETDEV\_FindCloseObjectSearchList](#_关闭目标检索结果查找,释放资源)接口，以释放资源；

#### 停止目标检索

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_StopObjectSearch  (  LPVOID lpUserID,  UINT32 udwSearchID  ); |

**接口描述：**

关闭目标检索结果查找,释放资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwSearchID | IN | 业务号 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 获取指定告警的关联数据

#### 获取指定告警的关联数据列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindAlarmRelatedDataList  (  LPVOID lpUserID,  INT32 dwAlarmID  ); |

**接口描述：**

获取指定告警的关联数据列表；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| dwAlarmID | IN | 告警ID |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextAlarmRelatedDataInfo](#_获取指定告警的关联数据)、

[NETDEV\_FindCloseAlarmRelatedDataList](#_关闭查询指定告警的关联数据)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextAlarmRelatedDataInfo](#_获取指定告警的关联数据)、[NETDEV\_FindCloseAlarmRelatedDataList](#_关闭查询指定告警的关联数据)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextAlarmRelatedDataInfo](#_获取指定告警的关联数据)接口获取告警关联数据；
* 获取告警关联数据结束后必须调用[NETDEV\_FindCloseAlarmRelatedDataList](#_关闭查询指定告警的关联数据)接口，以释放资源；

#### 获取指定告警的关联数据

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextAlarmRelatedDataInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_ALARM\_RELATED\_DATA\_S](#_告警关联数据) pstAlarmRelatedData  ); |

**接口描述：**

获取指定告警的关联数据；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_FindAlarmRelatedDataList](#_获取指定告警的关联数据列表)接口的返回值 |
| pstAlarmRelatedData | IN | 告警关联数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindAlarmRelatedDataList](#_获取指定告警的关联数据列表)、[NETDEV\_FindCloseAlarmRelatedDataList](#_关闭查询指定告警的关联数据)接口配套使用；
* 该接口需循环调用获取指定告警的关联数据；
* 获取告警关联数据结束后必须调用[NETDEV\_FindCloseAlarmRelatedDataList](#_关闭查询指定告警的关联数据)接口，以释放资源；

#### 关闭查询指定告警的关联数据

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseAlarmRelatedDataList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭查找指定告警的关联数据,释放资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_FindAlarmRelatedDataList](#_获取指定告警的关联数据列表)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindAlarmRelatedDataList](#_获取指定告警的关联数据列表)、[NETDEV\_FindNextAlarmRelatedDataInfo](#_获取指定告警的关联数据)接口配套使用；
* 获取告警关联数据结束后必须调用[NETDEV\_FindCloseAlarmRelatedDataList](#_关闭查询指定告警的关联数据)接口，以释放资源；

### 注册图片报警回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPicAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_PicAlarmMessCallBack\_PF cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册图片报警消息回调函数；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| cbAlarmMessCallBack | IN | 回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstAlarmPicData | IN | 告警图片信息 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 人数统计

#### 注册人数统计报警消息回调函数

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_SetPeopleCountAlarmCallBack  (  LPVOID lpUserID,  NETDEV\_PeopleCountAlarmMessCallBack\_PF cbAlarmMessCallBack,  LPVOID lpUserData  ); |

**接口描述：**

注册人数统计报警消息回调函数；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| cbAlarmMessCallBack | IN | 回调函数 |
| lpUserData | IN | 用户数据 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstAlarmInfo | IN | 告警图片信息 |
| lpUserData | IN | 用户数据 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 查询人员密度报警统计组列表信息

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindCrowdDensityGroupList  (  LPVOID lpUserID  ); |

**接口描述：**

查询人员密度报警统计组列表信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextCrowdDensityGroupInfo](#_逐个获取查找到的人员密度报警统计组信息)、

[NETDEV\_FindCloseCrowdDensityGroupList](#_逐个获取查找到的人员密度报警统计组信息)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextCrowdDensityGroupInfo](#_逐个获取查找到的人员密度报警统计组信息)、[NETDEV\_FindCloseCrowdDensityGroupList](#_逐个获取查找到的人员密度报警统计组信息)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextCrowdDensityGroupInfo](#_逐个获取查找到的人员密度报警统计组信息)接口获取人员密度报警统计组信息；
* 获取人员密度报警统计组信息结束后必须调用[NETDEV\_FindCloseCrowdDensityGroupList](#_逐个获取查找到的人员密度报警统计组信息)接口，以释放资源。

#### 逐个获取查找到的人员密度报警统计组信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextCrowdDensityGroupInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_CROWD\_DENSITY\_GROUP\_INFO\_S](#_人员密度报警统计组信息) pstCrowdDensityGroupInfo  ); |

**接口描述：**

查询目标检索结果；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_FindCrowdDensityGroupList](#_查询人员密度报警统计组列表信息)接口的返回值 |
| pstCrowdDensityGroupInfo | IN | 人员密度报警统计组信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindCrowdDensityGroupList](#_查询人员密度报警统计组列表信息)、[NETDEV\_FindCloseCrowdDensityGroupList](#_逐个获取查找到的人员密度报警统计组信息)接口配套使用；
* 该接口需循环调用获取人员密度报警统计组信息；
* 获取人员密度报警统计组信息结束后必须调用[NETDEV\_FindCloseCrowdDensityGroupList](#_逐个获取查找到的人员密度报警统计组信息)接口，以释放资源。

#### 关闭查询人员密度报警统计组列表信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseCrowdDensityGroupList  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭人员密度报警统计组信息查找,释放资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 查找句柄，[NETDEV\_FindCrowdDensityGroupList](#_查询人员密度报警统计组列表信息)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindCrowdDensityGroupList](#_查询人员密度报警统计组列表信息)、[NETDEV\_FindNextCrowdDensityGroupInfo](#_逐个获取查找到的人员密度报警统计组信息)接口配套使用；
* 获取人员密度报警统计组信息结束后必须调用[NETDEV\_FindCloseCrowdDensityGroupList](#_逐个获取查找到的人员密度报警统计组信息)接口，以释放资源；

#### 添加人员密度报警统计组信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

添加人员密度报警统计组信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstCrowdDensityGroupInfo | IN | 人员密度报警统计组信息 |
| pUdwGroupID | OUT | 统计组ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 修改人员密度报警统计组信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ModifyCrowdDensityGroupInfo  (  LPVOID lpUserID,  [LPNETDEV\_CROWD\_DENSITY\_GROUP\_INFO\_S](#_人员密度报警统计组信息) pstCrowdDensityGroupInfo  ); |

**接口描述：**

修改人员密度报警统计组信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstCrowdDensityGroupInfo | IN | 人员密度报警统计组信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 删除人员密度报警统计组信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_DeleteCrowdDensityGroupInfo  (  LPVOID lpUserID,  UINT32 udwCrowdDensityGroupID  ); |

**接口描述：**

删除人员密度报警统计组信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwCrowdDensityGroupID | IN | 人员密度报警统计组id |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 获取人员密度报警统计组信息

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取人员密度报警统计组信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwCrowdDensityGroupID | IN | 人员密度报警统计组id |
| pstCrowdDensityGroupInfo | OUT | 人员密度报警统计组信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

#### 执行绊线人数统计人数立即清零操作

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_ResetLinesPeopleCounting  (  LPVOID lpUserID,  UINT32 udwChannelID  ); |

**接口描述：**

执行绊线人数统计人数立即清零操作；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| udwChannelID | IN | 通道ID |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

## 系统维护

### 查找日志信息

#### 查找日志列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindLogInfoList  (  LPVOID lpUserID,  [LPNETDEV\_FIND\_LOG\_COND\_S](#_日志查询条件结构体) pstLogFindCond  ); |

**接口描述：**

根据日志类型、时间查找日志信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstLogFindCond | IN | 日志查询条件 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextLogInfo](#_逐个查找日志信息)、[NETDEV\_FindCloseLogInfo](#_关闭日志列表查找)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextLogInfo](#_逐个查找日志信息)、[NETDEV\_FindCloseLogInfo](#_关闭日志列表查找)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextLogInfo](#_逐个查找日志信息)接口获取日志信息；
* 获取日志信息结束后必须调用[NETDEV\_FindCloseLogInfo](#_关闭日志列表查找)接口，以释放资源,关闭查找；
* 此接口仅VMS及NVR支持，VMS仅支持操作日志查询；

**See Also：**

[NETDEV\_FindNextLogInfo](#_逐个查找日志信息)、[NETDEV\_FindCloseLogInfo](#_关闭日志列表查找)

#### 逐个查找日志信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextLogInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_LOG\_INFO\_S](#_日志信息结构体) pstLogInfo  ); |

**接口描述：**

逐个获取查找到的日志信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 日志查找句柄，[NETDEV\_FindLogInfoList](#_查找日志列表)接口的返回值 |
| pstLogInfo | OUT | 保存日志信息的指针 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindLogInfoList](#_查找日志列表)、[NETDEV\_FindCloseLogInfo](#_关闭日志列表查找)接口配套使用；
* 此接口需循环调用获取日志信息；
* 获取日志信息结束后必须调用[NETDEV\_FindCloseLogInfo](#_关闭日志列表查找)接口，以释放资源,关闭查找；
* 此接口仅VMS及NVR支持，VMS仅支持操作日志查询；

**See Also：**

[NETDEV\_FindLogInfoList](#_查找日志列表)、[NETDEV\_FindCloseLogInfo](#_关闭日志列表查找)

#### 关闭日志列表查找

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseLogInfo  (  LPVOID lpFindHandle  );s |

**接口描述：**

关闭日志查找，释放资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 日志查找句柄，[NETDEV\_FindLogInfoList](#_查找日志列表)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindLogInfoList](#_查找日志列表)、[NETDEV\_FindNextLogInfo](#_逐个查找日志信息)接口配套使用；
* 获取日志信息结束后必须调用[NETDEV\_FindCloseLogInfo](#_关闭日志列表查找)接口，以释放资源,关闭查找；

**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)

### 查询设备告警信息

#### 查找设备告警信息列表

**接口名称：**

|  |
| --- |
| LPVOID STDCALL NETDEV\_FindAlarmInfoList  (  LPVOID lpUserID,  [LPNETDEV\_FIND\_ALARM\_COND\_S](#_告警信息查找条件结构体) pstAlarmFindCond  ); |

**接口描述：**

根据时间查询设备告警；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstAlarmFindCond | IN | 告警查询条件 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息)、[NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息)、[NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息)接口获取设备告警信息；
* 获取告警信息结束后必须调用[NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找)接口，以释放资源,关闭查找；

**See Also：**

[NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息)、[NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找)

#### 逐个查找告警信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextAlarmInfo  (  LPVOID lpFindHandle,  [LPNETDEV\_FIND\_ALARM\_INFO\_S](#_告警信息结构体) pstAlarmInfo  ); |

**接口描述：**

逐个获取查找到的告警信息；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 告警信息查找句柄，[NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表)返回值 |
| pstAlarmInfo | OUT | 告警信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表)、[NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找)接口配套使用；
* 该接口需循环调用获取设备告警信息；
* 获取告警信息结束后必须调用[NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找)接口，以释放资源,关闭查找；

**See Also：**

[NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表)、[NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找)

#### 关闭告警信息查找

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseAlarmInfo  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭告警查询，释放资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 告警信息查找句柄，[NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表)返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表)、[NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息)接口配套使用；
* 获取告警信息结束后必须调用[NETDEV\_FindCloseAlarmInfo](#_关闭告警信息查找)接口，以释放资源,关闭查找；

**See Also：**

[NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表)、[NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息)

### 查询告警日志

#### 获取告警日志列表

**接口名称：**

|  |
| --- |
| 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  ); |

**接口描述：**

获取告警日志列表；

**Parameterss：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |
| pstFindCond | IN | 查询条件 |
| pdwTotalRealRow | OUT | 告警日志记录实际总条数 |

**Return Values：**

返回值为NULL则表示失败，其他值作为[NETDEV\_FindNextAlarmLog](#_逐个获取告警日志信息)、[NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找)入参；接口返回失败请调用[NETDEV\_GetLastError](#_获取接口错误码_1)获取错误码，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindNextAlarmLog](#_逐个获取告警日志信息)、[NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找)接口配套使用；
* 该接口调用成功后，循环调用[NETDEV\_FindNextAlarmLog](#_逐个获取告警日志信息)接口获取告警日志；
* 获取告警日志结束后必须调用[NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找)接口，以释放资源,关闭查找；
* 该接口仅VMS支持；

**See Also：**

[NETDEV\_FindNextAlarmLog](#_逐个获取告警日志信息)、[NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找)

#### 逐个获取告警日志信息

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindNextAlarmLog  (  LPVOID lpFindHandle,  LPNETDEV\_ALARM\_LOG\_INFO\_S pstAlarmLogInfo  ); |

**接口描述：**

逐个获取查找到的告警日志；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 告警信息查找句柄，[NETDEV\_FindAlarmLogList](#_获取告警日志列表)返回值 |
| pstAlarmLogInfo | OUT | 告警日志信息 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindAlarmLogList](#_获取告警日志列表)、[NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找)接口配套使用；
* 该接口需循环调用获取设备告警日志；
* 获取告警日志结束后必须调用[NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找)接口，以释放资源,关闭查找；

**See Also：**

[NETDEV\_FindAlarmLogList](#_获取告警日志列表)、[NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找)

#### 关闭告警日志查找

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_FindCloseAlarmLog  (  LPVOID lpFindHandle  ); |

**接口描述：**

关闭告警日志查找，释放资源；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpFindHandle | IN | 告警日志查找句柄，[NETDEV\_FindAlarmLogList](#_获取告警日志列表)返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 与[NETDEV\_FindAlarmLogList](#_获取告警日志列表)、[NETDEV\_FindNextAlarmLog](#_逐个获取告警日志信息)接口配套使用；
* 获取告警日志结束后必须调用[NETDEV\_FindCloseAlarmLog](#_关闭告警日志查找)接口，以释放资源,关闭查找。

### 设备重启

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_Reboot  (  PVOID lpUserID  ); |

**接口描述：**

重启设备；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

### 设备恢复出厂设置

**接口名称：**

|  |
| --- |
| BOOL STDCALL NETDEV\_RestoreConfig  (  LPVOID lpUserID  ); |

**接口描述：**

恢复出厂设置；

**Parameters：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpUserID | IN | 用户登录句柄，[NETDEV\_Login\_V30](#_用户设备登录)接口的返回值 |

**Return Values：**

TRUE表示成功,其他表示失败。获取错误码调用[NETDEV\_GetLastError](#_获取接口错误码_1)，通过错误码判断出错原因。

**Remarks**：

* 保留网络配置和用户配置,其他参数恢复到出厂设置

## 结构体定义

### 设备登录信息结构体

|  |
| --- |
| 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:**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szIPAddr | 设备IP地址/域名 |
| dwPort | 端口号，与web页面访问设备一致，默认为80端口 |
| szUserName | 用户名 |
| szPassword | 密码，用户输入明文，SDK内部进行加密 |
| dwLoginProto | 登录协议，参考枚举[NETDEV\_LOGIN\_PROTO\_E](#_接入协议枚举) |
| dwDeviceType | 设备类型，参考枚举[NETDEV\_DEVICE\_TYPE\_E](#_错误码列表) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_Login\_V30](#_用户设备登录)

### 安全登录信息结构体

|  |
| --- |
| typedef struct tagstNETDEVSELogInfo  {  INT32    dwSELogCount;  INT32    dwSELogTime;  BYTE   byRes[64];  }NETDEV\_SELOG\_INFO\_S, \*LPNETDEV\_SELOG\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSELogCount | 安全登录次数，如：用户名密码错误时，登录失败几次后，设备会被锁定 |
| dwSELogTime | 安全登录时间，如：设备被锁定后，多长时间才可尝试登录 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_Login\_V30](#_用户设备登录)

### 超时时间结构体

|  |
| --- |
| typedef struct tagNETDEVRevTimeout  {  INT32 dwRevTimeOut;  INT32 dwFileReportTimeOut;  BYTE byRes[128];  }NETDEV\_REV\_TIMEOUT\_S, \*LPNETDEV\_REV\_TIMEOUT\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwRevTimeOut | 网络消息超时时间，单位：秒 |
| dwFileReportTimeOut | 文件传输超时时间，单位：秒 |
| byRes | 保留字段 |

**Remarks：**

* 网络消息超时时间必须大于等于1秒；
* 文件传输超时时间必须大于等于30秒；

**See also：**

[NETDEV\_SetRevTimeOut](#_设置超时时间)

### 设备发现的设备信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szDevAddr | 设备地址 |
| szDevModule | 设备型号 |
| szDevSerailNum | 设备序列号 |
| szDevMac | 设备MAC地址 |
| szDevName | 设备名称 |
| szDevVersion | 设备版本 |
| enDevType | 设备类型，详见 [NETDEV\_DEVICE\_TYPE\_E](#_错误码列表) |
| dwDevPort | 设备端口号 |
| szManuFacturer | 生厂商 |
| szActiveCode | 激活码 |
| szCloudUserName | 云用户名称 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SetDiscoveryCallBack](#_设置设备搜索回调函数)

### 播放器异常输出信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwEventCode | 事件码（播放器上报异常使用） |
| tPlayBackTime | 回放绝对时间(播放器上报异常使用) |
| szFileName | 字符串参数 |
| dwOldSubID | 原有订阅ID，仅人脸订阅有用 |
| dwNewSubID | 新订阅ID，仅人脸订阅有用 |
| dwSubType | 订阅类型，仅人脸订阅有用 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SetExceptionCallBack](#_注册异常回调函数)

### 设备地址信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szUserName | 用户名 |
| szPassword | 密码 |
| szIPv4Address | Pv4的IP地址 |
| szIPv4GateWay | IPv4的网关地址 |
| szIPv4SubnetMask | IPv4的子网掩码 |
| szDevSerailNum | 设备序列号 |
| szDevMac | 设备MAC地址 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_ModifyDeviceAddr](#_修改设备IP地址)

### 实况预览参数结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道ID |
| dwStreamType | 码流类型，详见枚举 [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举) |
| dwLinkMode | 传输协议，详见枚举 [NETDEV\_PROTOCAL\_E](#_媒体传输协议枚举) |
| hPlayWnd | 播放窗口句柄 |
| dwFluency | 图像播放流程性优先类型，详见枚举 [NETDEV\_PICTURE\_FLUENCY\_E](#_图像播放流畅性枚举) |
| dwStreamMode | 起流模式，详见枚举 [NETDEV\_STREAM\_MODE\_E](#_起流模式枚举) |
| dwLiveMode | 起流模式，预留字段，传0即可； |
| dwDisTributeCloud | 分发能力，预留字段，传0即可； |
| dwallowDistribution | 是否支持分发，预留字段，传0即可； |
| dwTransType | 传输类型，仅VMS使用，详见枚举 [NETDEV\_TRANS\_TYPE\_E](#_传输类型枚举) |
| dwStreamProtocol | 起流协议，详见枚举 [NETDEV\_START\_STREAM\_PROT\_E](#_起流协议枚举) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_RealPlay](#_实时预览)

### 解码后音频数据结构体

|  |
| --- |
| ttypedef struct tagNETDEVWaveData  {  CHAR \*pcData;  INT32 dwDataLen;  INT32 dwWaveFormat;  }NETDEV\_WAVE\_DATA\_S, \*LPNETDEV\_WAVE\_DATA\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| pcData | 音频数据 |
| dwDataLen | 音频数据长度 |
| dwWaveFormat | 解码后音频格式，详见枚举[NETDEV\_WAVE\_FORMAT\_INFO\_E](#_解码后音频格式) |

**See also：**

[NETDEV\_StartVoiceCom](#_开启语音对讲)

### 音频参数结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| ulChannels | 声道数，单声道为，立体声为 |
| ulSampleRate | 采样率 |
| enSampleFormat | 位宽 |

**See also：**

[NETDEV\_InputVoiceData](#_语音数据转发)

### 通道操作信息结构体

|  |
| --- |
| typedef struct tagNETDEVOperateInfo  {  INT32 dwID;  INT32 dwReturnCode;  BYTE byRes[64];  }NETDEV\_OPERATE\_INFO\_S, \*LPNETDEV\_OPERATE\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwID | 操作ID |
| dwReturnCode | 返回码 |
| byRes | 保留字段 |

### 通道列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 批量处理数量 |
| pstOperateInfo | 批量处理信息 |

**See also：**

[NETDEV\_CreateVoiceBroadcastGroup](#_创建语音广播组)

### 语音广播组批处理信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| pstAddChannelList | 新增通道列表 |
| pstDelChannelList | 删除通道列表 |

**See also：**

[NETDEV\_ModifyVoiceBroadcastGroup](#_修改语音广播组)

### 语音广播通道信息结构体

|  |
| --- |
| typedef struct tagstNETDEV\_VoiceBroadcastChlInfo  {  UINT32 dwChannelID;  UINT32 dwResultCode;  UINT32 dwStatus;  }NETDEV\_VOICE\_BROADCAST\_CHL\_INFO\_S, \*LPNETDEV\_VOICE\_BROADCAST\_CHL\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 音频通道ID |
| dwResultCode | 广播通道状态码 |
| dwStatus | 广播通道声音开启状态 |

### 语音广播组通道信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 广播组通道个数 |
| astChlInfoList | 所有通道信息 |

**See also：**

[NETDEV\_GetVoiceBroadcastChlStatus](#_获取语音广播组通道状态)

### 语音广播组控制信息结构体

|  |
| --- |
| typedef struct tagstNETDEV\_VoiceBroadcastCtrlInfo  {  INT32 dwChannelID;  UINT32 udwStatus;  }NETDEV\_VOICE\_BROADCAST\_CTRL\_INFO\_S, \*LPNETDEV\_VOICE\_BROADCAST\_CTRL\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道ID |
| udwStatus | 声音状态,开(1),关(0) |

**See also：**

[NETDEV\_ModifyVoiceBroadcastStatus](#_修改语音广播组通道状态)

### 音频数据结构体

|  |
| --- |
| typedef struct tagNETDEVWaveData  {  CHAR \*pcData;  INT32 dwDataLen;  INT32 dwWaveFormat;  }NETDEV\_WAVE\_DATA\_S, \*LPNETDEV\_WAVE\_DATA\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| pcData | 音频数据 |
| dwDataLen | 音频数据长度 |
| dwWaveFormat | 解码后音频格式，详见枚举 [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)

### 图像数据结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| pucData | YUV数据， pucData[0]:Y 平面指针,pucData[1]:U 平面指针,pucData[2]:V 平面指针 |
| dwLineSize | dwLineSize [0]:Y平面每行跨距, dwLineSize [1]:U平面每行跨距, dwLineSize [2]:V平面每行跨距 |
| dwPicHeight | 图片高度 |
| dwPicWidth | 图片宽度 |
| dwRenderTimeType | 用于渲染的时间数据类型 |
| tRenderTime | 用于渲染的时间数据 |

**See also：**

[NETDEV\_SetPlayDecodeVideoCB](#_注册解码后视频数据回调)

### 拼帧后音频数据结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| pucData | 音频数据 |
| udwDataLen | 音频数据长度 |
| udwAudioCodeFormat | 音频编码格式，详见枚举 [NETDEV\_AUDIO\_ENCODE\_TYPE\_E](#_音频编码格式枚举) |
| udwWaveFormat | 音频数据解码后音频格式，详见枚举 [NETDEV\_WAVE\_FORMAT\_INFO\_E](#_解码后音频格式) |
| tTimeStamp | 时间戳 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SetPlayParseAudioCB](#_注册拼帧后音频数据回调)

### 视频解析数据结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| pucData | 视频数据 |
| dwDataLen | 视频数据长度 |
| dwVideoFrameType | 视频帧类型，详见枚举 [NETDEV\_VIDEO\_FRAME\_TYPE\_E](#_视频帧类型枚举) |
| dwVideoCodeFormat | 视频编码格式，详见枚举 [NETDEV\_VIDEO\_CODE\_TYPE\_E](#_视频编码格式枚举) |
| dwHeight | 视频图像高度 |
| dwWidth | 视频图像宽度 |
| tTimeStamp | 时间戳（毫秒） |
| tAbTime | 绝对时间(unix时间戳)，当前仅回放流存在 |

**See also：**

[NETDEV\_SetPlayParseCB](#_注册拼帧后视频数据回调)

### 影像信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwContrast | 对比度 |
| dwBrightness | 亮度 |
| dwSaturation | 饱和度 |
| dwHue | 色调 |
| dwGamma | 伽玛值 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_GetVideoEffect](#_获取影像参数)、[NETDEV\_SetVideoEffect](#_影像调节)

### 矩形区域结构体

|  |
| --- |
| typedef struct tagNETDEVRect  {  INT32 dwLeft;  INT32 dwTop;  INT32 dwRight;  INT32 dwBottom;  }NETDEV\_RECT\_S, \*LPNETDEV\_RECT\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwLeft | x轴左点值[0,10000] |
| dwTop | y轴顶点值[0,10000] |
| dwRight | x轴右点值[0,10000] |
| dwBottom | y轴底点值[0,10000] |

**See also：**

[NETDEV\_SetDigitalZoom](#_设置数字放大)

### 录像查找信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szFileName | 录像文件名 |
| dwChannelID | 通道号 |
| dwStreamType | 码流类型，详见枚举 [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举) |
| dwFileType | 录像存储类型，详见枚举 [NETDEV\_STORE\_TYPE\_E](#_录像存储类型枚举) |
| tBeginTime | 起始时间 |
| tEndTime | 结束时间 |
| dwRecordLocation | 录像存储位置，详见枚举 [NETDEV\_RECORD\_LOCATION\_E](#_录像存储位置枚举) |
| udwServerID | 录像所属服务器ID，仅VMS使用 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_FindFile](#_查找录像文件列表)

### 录像文件信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szFileName | 录像文件名 |
| tBeginTime | 起始时间 |
| tEndTime | 结束时间 |
| byFileType | 录像存储类型，详见枚举 [NETDEV\_STORE\_TYPE\_E](#_录像存储类型枚举) |
| udwServerID | 录像所属服务器ID，仅VMS使用 |
| udwFileSize | 录像文件大小 |
| dwFileType | 文件类型，参考[NETDEV\_RECORD\_SEARCH\_TYPE\_E](#_按位查询录像类型枚举)，暂内部使用 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_FindNextFile](#_逐个获取查找到的文件信息)

### 按月查询录像分布信息结构体

|  |
| --- |
| typedef struct tagNETDEVMonthInfo  {  UINT32 udwYear;  UINT32 udwMonth;  UINT32 udwPosition;  }NETDEV\_MONTH\_INFO\_S, \*LPNETDEV\_MONTH\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwYear | 年份 |
| udwMonth | 月份 |
| udwPosition | 录像查询位置，详见枚举 [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)

### 录像分布状态结构体

|  |
| --- |
| typedef struct tagNETDEVMonthStatus  {  UINT32 udwDayNumInMonth;  UINT32 szVideoStatus[NETDEV\_MONTH\_DAY\_MAX];  }NETDEV\_MONTH\_STATUS\_S, \*LPNETDEV\_MONTH\_STATUS\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwDayNumInMonth | 某月的实际天数 |
| szVideoStatus | 录像状态列表，详见枚举 [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)

### 通道列表结构体

|  |
| --- |
| typedef struct tagNETDEVChannels  {  UINT32 udwSize;  INT32 dwChannelIDs[NETDEV\_CHANNEL\_MAX];  }NETDEV\_CHANNELS\_S, \*LPNETDEV\_CHANNELS\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSize | 通道个数 |
| dwChannelIDs | 需要查询的视频输入通道号列表 |

**See also：**

[NETDEV\_QuickSearch](#_单通道查询)、[NETDEV\_QuickSearchMultiChl](#_多通道查询)

### 按时间回放录像参数结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道号 |
| tBeginTime | 开始时间 |
| tEndTime | 结束时间 |
| dwLinkMode | 传输协议，详见枚举 [NETDEV\_PROTOCAL\_E](#_媒体传输协议枚举) |
| hPlayWnd | 播放窗口句柄 |
| dwFileType | 录像存储类型，详见枚举 [NETDEV\_STORE\_TYPE\_E](#_录像存储类型枚举) |
| dwDownloadSpeed | 下载速度，录像下载时有效，详见枚举 [NETDEV\_E\_DOWNLOAD\_SPEED\_E](#_录像下载速度枚举) |
| dwStreamMode | 起流模式，详见枚举 [NETDEV\_STREAM\_MODE\_E](#_起流模式枚举) |
| dwStreamIndex | 码流类型，详见枚举 [NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举) |
| dwRecordLocation | 录像存储位置，仅VMS支持，详见枚举 [NETDEV\_RECORD\_LOCATION\_E](#_录像存储位置枚举) |
| dwTransType | 传输类型，仅VMS使用，详见枚举 [NETDEV\_TRANS\_TYPE\_E](#_传输类型枚举) |
| bCloudStorage | 是否开启云存储回放模式，预留参数，仅限内部使用； |
| bOneFrameEnable | 是否开启单帧解码模式，开启后对解码效率有影响 |
| dwPlaySpeed | 回放播放速度，详见枚举 [NETDEV\_VOD\_PLAY\_STATUS\_E](#_播放/下载速度枚举) |
| cbPlayDecodeVideoCALLBACK | 解码后数据回调函数 |
| tPlayTime | 播放时间 |
| udwServerID | 录像所属服务器ID，仅VMS使用 |
| byRes | 保留字段 |

**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**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 传参说明 |
| lpPlayHandle | IN | 当前实时预览句柄/回放句柄 |
| pstPictureData | IN | 存放解码后视频流数据信息缓冲区，详见结构体[NETDEV\_PICTURE\_DATA\_S](#_媒体数据流格式) |
| lpUserParam | IN | 用户数据，在调用NETDEV\_SetPlayDecodeVideoCB时指定的用户参数 |

**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)

### 按文件名回放录像参数结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szName | 回放的控制块名称 |
| tBeginTime | 开始时间 |
| tEndTime | 结束时间 |
| dwLinkMode | 传输协议，详见枚举 [NETDEV\_PROTOCAL\_E](#_媒体传输协议枚举) |
| hPlayWnd | 播放窗口句柄 |
| dwFileType | 录像存储类型，保留参数，暂未使用，默认0即可 |
| dwDownloadSpeed | 下载速度，录像下载时有效，详见枚举 [NETDEV\_E\_DOWNLOAD\_SPEED\_E](#_录像下载速度枚举) |
| dwStreamMode | 起流模式，详见枚举 [NETDEV\_STREAM\_MODE\_E](#_起流模式枚举) |
| byRes | 保留字段 |

**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)

### 拉框缩放结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwMidPointX | 拉框中心横坐标 |
| udwMidPointY | 拉框中心纵坐标 |
| udwLengthX | 拉框长度 |
| udwLengthY | 拉框宽度 |
| udwWidth | 实际播放窗口长度 |
| udwHeight | 实际播放窗口宽度 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_PTZ3DPosition](#_云台3D定位_1)

### 云台区域信息结构体

|  |
| --- |
| typedef struct tagNETDEVPtzOperateAreaInfo  {  INT32 dwBeginPointX;  INT32 dwBeginPointY;  INT32 dwEndPointX;  INT32 dwEndPointY;  }NETDEV\_PTZ\_OPERATEAREA\_S, \*LPNETDEV\_PTZ\_OPERATEAREA\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwBeginPointX | 区域起始点X值[0,10000] |
| dwBeginPointY | 区域起始点Y值[0,10000] |
| dwEndPointX | 区域结束点X值[0,10000] |
| dwEndPointY | 区域结束点Y值[0,10000] |

**See also：**

[NETDEV\_PTZSelZoomIn](#_拉框放大.缩小_1)

### 云台守望信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 云台守望使能 |
| udwMode | 云台守望模式。参考枚举[NETDEV\_PTZ\_GUARD\_MODE\_E](#_云台守望模式) |
| udwParam | 参数,当Mode为0时，Param表示预置位序号，从1开始; 当Mode为1时，Param表示巡航线路索引，从0开始;当Mode为2时，Param表示轨迹巡航线路索引，0开始 |
| udwTime | 守望时间，云台静默此时间后自动守望，单位s |
| byRes | 保留字段 |

**See also：**

[NETDEV\_PTZGetGuard\_Other](#_获取、设置云台守望信息)

### 绝对坐标移动

|  |
| --- |
| typedef struct tagNETDEVPTZAbsoluteMove  {  FLOAT fPanTiltX;  FLOAT fPanTiltY;  FLOAT fZoomX;  }NETDEV\_PTZ\_ABSOLUTE\_MOVE\_S, \*LPNETDEV\_PTZ\_ABSOLUTE\_MOVE\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| fPanTiltX | 绝对水平坐标 |
| fPanTiltY | 绝对竖直坐标 |
| fZoomX | 绝对聚焦倍数 |

**See also：**

[NETDEV\_PTZAbsoluteMove](#_云台绝对坐标转动)

### 云台状态

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| fPanTiltX | 绝对水平坐标 |
| fPanTiltY | 绝对竖直坐标 |
| fZoomX | 绝对聚焦倍数 |
| enPanTiltStatus | 云台状态 |
| enZoomStatus | 聚焦状态 |

**See also：**

[NETDEV\_PTZGetStatus](#_获取云台状态)

### 云台轨迹巡航路径结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwTrackNum | 已存在的巡航轨迹的数量 |
| aszTrackName | 轨迹名称 |

**See also：**

[NETDEV\_PTZGetTrackCruise](#_获取云台轨迹巡航路径_1)

### 云台轨迹巡航路径列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwTrackNum | 已存在的巡航轨迹的数量 |
| astTrackInfo | 轨迹名称 |

**See also：**

[NETDEV\_PTZGetTrackCruise\_V30](#_获取云台轨迹巡航路径)

### 云台轨迹巡航路径结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTrackID | 巡航轨迹的ID |
| szTrackName | 轨迹名称 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_PTZTrackCruise\_V30](#_云台轨迹巡航操作)

### 云台轨迹巡航状态信息结构体

|  |
| --- |
| typedef struct tagNETDEVPTZTrackStatusInfo  {  INT32 dwStatus;  INT32 dwPatrolID;  }NETDEV\_PTZ\_TRACK\_STATUS\_INFO\_S, \*LPNETDEV\_PTZ\_TRACK\_STATUS\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwStatus | 通道巡航状态。参考[NETDEV\_PTZ\_TRACK\_STATUS\_E](#_巡航状态枚举) |
| dwPatrolID | 录制或者巡航路线序号 |

**See also：**

[NETDEV\_PTZGetTrackStatus](#_获取指定通道的巡航状态)

### 所有云台预置位结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 预置位总数 |
| astPreset | 预置位信息结构体 |

**See also：**

[NETDEV\_GetPTZPresetList](#_获取云台预置位列表)

### 云台预置位信息结构体

|  |
| --- |
| typedef struct tagstNETDEVPtzPreset  {  INT32 dwPresetID;  CHAR szPresetName[NETDEV\_LEN\_32];  }NETDEV\_PTZ\_PRESET\_S, \*LPNETDEV\_PTZ\_PRESET\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwPresetID | 预置位ID |
| szPresetName | 预置位名称 |

**See also：**

[NETDEV\_PTZ\_ALLPRESETS\_S](#_获取云台预置位列表)

### 云台预置位巡航路径详细信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwCuriseID | 轨迹ID |
| szCuriseName | 轨迹名称 |
| dwSize | 路径包含的轨迹点数量 |
| astCruisePoint | 路径包含的轨迹点信息 |

**See also：**

[NETDEV\_PTZCruise](#_云台预置位巡航操作_1)

### 云台预置位巡航轨迹点结构体

|  |
| --- |
| typedef struct tagNETDEVCruisePoint  {  INT32 dwPresetID;  INT32 dwStayTime;  INT32 dwSpeed;  INT32 dwID;  }NETDEV\_CRUISE\_POINT\_S, \*LPNETDEV\_CRUISE\_POINT\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwPresetID | 预置位ID |
| dwStayTime | 停留时间 |
| dwSpeed | 转动速度,范围[1-10] |
| dwID | 巡航动作ID |

**See also：**

[NETDEV\_CRUISE\_INFO\_S](#_云台预置位巡航路径详细信息结构体)

### 云台预置位巡航路径列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 巡航路径数量 |
| astCruiseInfo | 巡航路径信息 |

**See also：**

[NETDEV\_PTZGetCruise](#_获取云台预置位巡航路径)

### 云台方位标定信息结构体

|  |
| --- |
| typedef struct tagNETDEVPTZOrientationInfo  {  INT32 dwDirection;  BYTE byRes[252];  }NETDEV\_PTZ\_ORIENTATION\_INFO\_S, \*LPNETDEV\_PTZ\_ORIENTATION\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwDirection | 云台方位信息。参加枚举[NETDEV\_PTZ\_DIRECTION\_E](#_云台方位) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_PTZCalibrate](#_云台标定)

### 云台镜头当前视场角结构体

|  |
| --- |
| typedef struct tagNETDEVPTZLensAngle  {  FLOAT fHorizontalAngle;  FLOAT fVerticalAngle;  BYTE byRes[256];  }NETDEV\_PTZ\_LENSANGLE\_S, \*LPNETDEV\_PTZ\_LENSANGLE\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| fHorizontalAngle | 水平视场角取值范围[0.180] |
| fVerticalAngle | 垂直视场角取值范围[0.180] |
| byRes | 保留字段 |

**See also：**

[NETDEV\_PTZGetLensAngle](#_获取指定视频输入通道的云台镜头当前视角值)

### 云台经纬度信息单位

|  |
| --- |
| typedef struct tagNETDEVPTZPtPositionInfo  {  FLOAT fLongitude;  FLOAT fLatitude;  BYTE byRes[128];  }NETDEV\_PTZ\_PT\_POSITION\_INFO\_S, \*LPNETDEV\_PTZ\_PT\_POSITION\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| fLongitude | 云台经度（云台水平移动角度）范围：[0.00, 360.00] （单位:度精确到小数点后两位） |
| fLatitude | 云台纬度（云台上下翻转角度） |
| byRes | 保留字段 |

**See also：**

[NETDEV\_GetPTZAbsolutePTInfo](#_获取云台的经纬度信息)

### 日志查询条件结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwMainType | 日志主类型，详见枚举 [NETDEV\_LOG\_MAIN\_TYPE\_E](#_日志主类型枚举) |
| dwSubType | 日志子类型，详见枚举 [NETDEV\_LOG\_SUB\_TYPE\_E](#_日志子类型) |
| tBeginTime | 起始时间 |
| tEndTime | 结束时间 |
| dwLimitNum | 每次查询的数量 |
| dwOffset | 从序号几开始查询，序号从0开始 |

**See also：**

[NETDEV\_FindLogInfoList](#_查找日志列表)

### 日志信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| tTime | 日志时间 |
| dwMainType | 日志主类型，详见枚举 [NETDEV\_LOG\_MAIN\_TYPE\_E](#_日志主类型枚举) |
| dwSubType | 日志子类型，详见枚举 [NETDEV\_LOG\_SUB\_TYPE\_E](#_日志子类型) |
| dwChannelID | 日志来源（通道号） |
| szUserName | 用户名 |
| szUserAddr | 用户IP地址 |
| szDetailInfo | 详细信息，仅NVR支持 |
| szOperObject | 操作对象，仅VMS支持 |
| udwOperResult | 操作结果，仅VMS支持 |
| szDevName | 设备名称 |
| szOrgName | 组织名称 |
| bRes | 保留字段 |

**See also：**

[NETDEV\_FindNextLogInfo](#_逐个查找日志信息)

### 告警信息查找条件结构体

|  |
| --- |
| typedef struct tagNETDEVAlarmFindConds  {  INT32 dwChannelID;  INT64 tBeginTime;  INT64 tEndTime;  BYTE byRes[128];  }NETDEV\_FIND\_ALARM\_COND\_S, \*LPNETDEV\_FIND\_ALARM\_COND\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道号 |
| tBeginTime | 开始时间 |
| tEndTime | 结束时间 |
| bRes | 保留字段 |

**See also：**

[NETDEV\_FindAlarmInfoList](#_查找设备告警信息列表)

### 告警信息结构体

|  |
| --- |
| typedef struct tagNETDEVFindAlarmInfo  {  INT64 tAlarmTime;  INT32 dwAlarmType;  BYTE byRes[256];  }NETDEV\_FIND\_ALARM\_INFO\_S, \*LPNETDEV\_FIND\_ALARM\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| tAlarmTime | 告警时间 |
| dwAlarmType | 告警类型，详见枚举 [NETDEV\_FIND\_ALARM\_TYPE\_E](#_告警类型枚举) |
| bRes | 保留字段 |

**See also：**

[NETDEV\_FindNextAlarmInfo](#_逐个查找告警信息)

### 告警日志查询条件列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwPageRow | 每页最大条数 |
| dwFirstRow | 分页查询中第一条数据的序号 |
| dwCondSize | 查询条件数量 |
| astCondition | 查询条件右值 |

**See also：**

[NETDEV\_FindAlarmLogList](#_获取告警日志列表)、[NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录)

### 告警日志查询条件结构体

|  |
| --- |
| typedef struct tagstNETDEVQueryInfo  {  INT32 dwQueryType;  INT32 dwLogicFlag;  CHAR szConditionData[NETDEV\_CODE\_STR\_MAX\_LEN];  }NETDEV\_QUERY\_INFO\_S, \*LPNETDEV\_QUERY\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwQueryType | 查询条件类型，详见枚举 [NETDEV\_QUERYCOND\_TYPE\_E](#_告警日志查询条件枚举) |
| dwLogicFlag | 查询条件逻辑类型，详见枚举[NETDEV\_QUERYCOND\_LOGICTYPE\_E](#_查询条件逻辑类型枚举) |
| szConditionData | 查询条件右值 |

**See also：**

[NETDEV\_ALARM\_LOG\_COND\_LIST\_S](#_告警日志查询信息结构体)

### 告警日志信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwAlarmID | 告警ID |
| dwAlarmType | 告警类型，详见枚举 [NETDEV\_ALARM\_TYPE\_E](#_告警类型枚举_1) |
| dwAlarmSubType | 告警子类型，详见枚举[NETDEV\_ALARM\_SUBTYPE\_E](#_告警子类型枚举) |
| dwAlarmLevel | 告警等级 0：紧急 1：重要 2：次要 3：警告 4：提示 |
| dwServerID | 所属服务器 |
| dwDevID | 设备ID |
| dwChannelID | 通道ID |
| szAlarmSrc | 告警源信息 |
| tAlarmTime | 告警发生时间UTC时间格式，单位为秒 |
| bAlarmChecked | 告警是否被确认，：未确认1：确认 |
| szAlarmCheckUser | 告警确认用户 |
| tAlarmCheckTime | 告警确认时间UTC时间格式，单位为秒 |
| szAlarmCheckDesc | 告警确认描述 |
| dwAlarmLinkType | 告警联动类型 |
| IsAlarmSnapExisted | 告警是否有抓图0没有抓图1有抓图 |
| dwAlarmSrcBelong | 告警所属，详见枚举[NETDEV\_ALARM\_SRC\_BELONG\_TYPE\_E](#_告警所属类型枚举) |
| szAlarmDetail | 告警详情，补充业务层需要展示的信息 |
| dwHasRelatedData | 告警是否存在关联数据 0：不存在 1：存在 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_FindNextAlarmLog](#_逐个获取告警日志信息)

### 人员信息列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 人员库人员个数 |
| pstPersonInfo | 人员信息列表,需动态分配内存 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_AddPersonInfo](#_新增指定的人员信息)

### 人员信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPersonID | 人员ID |
| udwLastChange | 人员信息最后修改时间 |
| szPersonName | 人员名字 范围:[1, 63] |
| udwGender | 成员性别 详情参见枚举[NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举) |
| szBirthday | 成员出生日期，格式YYYYMMDD，范围[1,31] |
| stRegionInfo | 成员地区信息 |
| udwTimeTemplateNum | 时间模板数量 |
| pstTimeTemplateList | 时间模板相关信息 需动态分配内存 |
| udwIdentificationNum | 证件信息个数 范围:[0, 6] |
| stIdentificationInfo | 成员证件信息 |
| udwImageNum | 人脸图片个数 范围:[0, 6] |
| stImageInfo | 人脸图片信息列表 |
| udwReqSeq | 请求数据序列号，此字段会在返回结果中待会，仅在批量添加中携带该字段 |
| bIsMonitored | 是否已布控，获取时必带，仅VMS支持 |
| udwBelongLibNum | 所属人员库数量，仅VMS支持 |
| pudwBelongLibList | 所属人员库ID列表，需动态分配内存，仅VMS支持 |
| udwCustomNum | 自定义属性值数量，最多5个，仅VMS支持 |
| pstCustomValueList | 自定义属性值列表，需动态分配内存，当Num为0时可以不填 |
| szTelephone | 联系电话 字符串长度[1,64] |
| szAddress | 地址 字符串长度范围[1, 64] |
| udwCardNum | 门禁卡个数 取值范围[0,6],Get时携带 |
| udwFingerprintNum | 指纹个数，取值范围[0,10] |
| udwType | 人员类型 0：员工 1：访客 2：陌生人 |
| stStaff | 员工信息 |
| stVisitor | 访客信息 |
| szDesc | 备注信息 字符串长度范围[1, 128] |
| szPersonCode | 人员编码，可填写学号或工号，范围:[1, 15] PTS支持 |
| szRemarks | 备注信息 范围:[1-63] PTS支持 |
| byRes | 保留字段 |

### 成员地区信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szNation | 国籍，范围[1-63] |
| szProvince | 省份，范围[1-63] |
| szCity | 城市，范围[1-63] |
| byRes | 保留字段 |

### 时间模板相关信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwBeginTime | 时间模板生效起始时间 若未配置，填写0 |
| udwEndTime | 时间模板生效结束时间 若未配置，填写4294967295(0xFFFFFFFF) |
| udwIndex | 时间模板索引 |
| byRes | 保留字段 |

### 成员证件信息结构体

|  |
| --- |
| typedef struct tagNETDEVIdentificationInfo  {  UINT32 udwType;  CHAR szNumber[NETDEV\_LEN\_128];  BYTE byRes[32];  }NETDEV\_IDENTIFICATION\_INFO\_S,\*LPNETDEV\_IDENTIFICATION\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 证件类型 详情参见枚举[NETDEV\_ID\_TYPE\_E](#_证件类型枚举) |
| szNumber | 证件号，范围:[1, 127] |
| byRes | 保留字段 |

### 人脸图片信息列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwFaceID | 人脸照片ID |
| stFileInfo | 文件信息 |
| udwModelStatus | 建模状态, 详情参见枚举[NETDEV\_MODEL\_STATUS\_E](#_建模状态枚举) |
| byRes | 保留字段 |

### 文件信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szName | 文件名称 范围[1, 16] |
| udwSize | 文件大小[data或通过szurl获取到的图片大小(Base64编码后)] |
| dwFileType | 文件类型，详情参见枚举[NETDEV\_FILE\_TYPE\_E](#_文件信息枚举) |
| udwLastChange | 最后修改时间，UTC时间，单位为s |
| pcData | 文件数据 Base64 需根据udwSize 动态申请内存 |
| szUrl | 图片URL，长度范围[0,256] |
| byRes | 保留字段 |

### 自定义属性信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 自定义属性名称序号 从0开始 |
| szValue | 自定义属性值 范围[1,63] |
| udwModelStatus | 建模状态, 详情参见枚举[NETDEV\_MODEL\_STATUS\_E](#_建模状态枚举) |
| byRes | 保留字段 |

### 员工信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szNumber | 人员编号 字符串长度范围[1, 16] |
| szBirthday | 出生日期 字符串长度范围[1,31] |
| udwDeptID | 部门ID |
| szDeptName | 部门名称 添加时可不携带 字符串长度范围[1, 64] |
| byRes | 保留字段 |

### 访客信息结构体

|  |
| --- |
| typedef struct tagNETDEVVisitorInfo  {  UINT32 udwVisitorCount;  CHAR szCompany[NETDEV\_LEN\_256];  UINT32 udwIntervieweeID;  BYTE byRes[256];  }NETDEV\_VISITOR\_INFO\_S,\*LPNETDEV\_VISITOR\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwVisitorCount | 访客人数 |
| szCompany | 访客单位 字符串长度范围[1, 64] |
| udwIntervieweeID | 被访者ID |
| byRes | 保留字段 |

### 人员信息结果列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 人员个数 |
| pstPersonList | 人员信息执行结果列表,需动态分配内存 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_AddPersonInfo](#_新增指定的人员信息)

### 人员信息执行结果结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPersonID | 人员ID |
| udwFaceNum | 人脸个数 批量单次最多6个 |
| stFaceInfo | 人脸信息结果列表 |
| udwReqseq | 请求数据序号,仅VMS支持 |
| byRes | 保留字段 |

### 人脸信息结果结构体

|  |
| --- |
| typedef struct tagNETDEVFaceInfo  {  UINT32 udwFaceID;  UINT32 udwResultCode;  BYTE byRes[128];  }NETDEV\_FACE\_INFO\_S,\*LPNETDEV\_FACE\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwFaceID | 人员ID |
| udwResultCode | 处理结果状态码，详情参见枚举[NETDEV\_PERSON\_RESULT\_CODE\_E](#_人脸处理结果状态码枚举) |
| byRes | 保留字段 |

### 布控任务信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 人脸布控任务序号 添加布控不再返回该字段 |
| stMonitorRuleInfo | 人脸布控任务配置信息 |
| udwLinkStrategyNum | 告警联动策略数量 |
| pstLinkStrategyList | 告警联动策略配置信息，需动态分配内存 |
| stWeekPlan | 人脸布控任务布防计划,仅NVR IPC使用 |
| stMonitorDefenceInfo | 布防信息，仅PTS VMS使用 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_AddPersonMonitorInfo](#_新增单个人脸布控任务)

### 布控任务配置信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 布控任务使能 |
| szName | 布控任务的布控名称 |
| szReason | 布控的布控原因 |
| udwLibNum | 任务对应的库数量,最大16个 |
| audwLibList | 库ID列表 |
| udwMonitorType | 布控告警类型，0：匹配告警,1：不匹配告警 |
| udwMultipleValue | 人脸1：N比较置信度阀值 |
| udwMonitorReason | 车辆布控原因：0：被抢车 1：被盗车 2：嫌疑车 3：交通违法车 4：紧急查控车 |
| szMatchSucceedMsg | 比对成功提示语 |
| szMatchFailedMsg | 比对失败提示语 |
| udwMemberNum | 成员数量 [0-32](NVR 不支持) |
| stMemberInfo | 成员信息列表 |
| udwChannelNum | 人脸布控任务的布控通道个数 获取单个布控任务详情时必带 |
| pudwMonitorChlIDList | 人脸布控任务的布控通道列表 根据udwChannelNum动态确定上层申请内存 |
| udwDevNum | 布控任务的布控设备个数 最多四个,仅VMS支持 |
| audwMonitorDevIDList | 布控任务的布控设备列表 根据DevNum动态确定,仅VMS支持 |
| udwMonitorRuleType | 布控任务智能类型，0：本地智能布控,1：前端智能布控 |
| byRes | 保留字段 |

### 人脸/车辆成员信息列表结构体

|  |
| --- |
| typedef struct tagNETDEVMemberInfo  {  UINT32 udwMemberID;  CHAR szMemberName[NETDEV\_LEN\_260];  UINT32 udwStatus;  BYTE byRes[124];  }NETDEV\_MEMBER\_INFO\_S,\*LPNETDEV\_MEMBER\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwMemberID | 人脸/车辆成员ID |
| szMemberName | 人脸/车辆成员名称，范围[1,63] |
| udwStatus | 成员同步状态 人脸参考：[NETDEV\_PERSON\_RESULT\_CODE\_E](#_人脸处理结果状态码枚举) |
| byRes | 保留字段 |

### 告警联动配置信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 告警类型，详情参见枚举[NETDEV\_PERSON\_COMPARE\_RESULT\_TYPE\_E](#_告警联动配置信息枚举) |
| stLintageActions | 人脸布控任务联动动作 |
| byRes | 保留字段 |

### 布控任务联动动作列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 联动动作列表数量 |
| stActionInfo | 联动动作列表信息 |
| byRes | 保留字段 |

### 联动动作列表信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwActID | 联动动作ID，详情参见枚举[NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举) |
| stEnabledInfo | 联动参数使能标记，适用于联动蜂鸣器、联动EMail、联动告警弹窗 |
| stOutputSwitchActParamInfo | 联动开关量输出 |
| stChannelActParamInfo | 联动NVR预览、联动存储、联动抓拍 |
| stPresetActParamInfo | 联动云台预置位 |
| byRes | 保留字段 |

### 通道联动信息结构体

|  |
| --- |
| typedef struct tagNETDEVChannelActParamInfo  {  UINT32 udwNum;  INT32 adwChannelID[NETDEV\_CHANNEL\_MAX];  }NETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S,\*LPNETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 通道个数 |
| adwChannelID | 通道ID列表 |

### 联动使能参数结构体

|  |
| --- |
| typedef struct tagNETDEVEnabledActParamInfo  {  BOOL bEnabled;  BYTE byRes[64];  }NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S, \*LPNETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 使能标记 |
| byRes | 保留字段 |

### 联动云台预置位列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 联动动作数量 |
| stChannelPreset | 联动到预置位信息列表 |

### 联动云台预置位信息结构体

|  |
| --- |
| typedef struct tagNETDEVChannelPreset  {  INT32 dwChannelID;  INT32 dwPresetID;  BYTE byRes[128];  }NETDEV\_CHANNEL\_PRESET\_S,\*LPNETDEV\_CHANNEL\_PRESET\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道号 |
| dwPresetID | 预置位编号 |
| byRes | 保留字段 |

### 联动开关量输出结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 联动的开关量输出个数 |
| astOutputAlarmStatusInfo | 联动的开关量输出列表 |

### 输出开关量的逻辑报警状态(手动告警)结构体

|  |
| --- |
| typedef struct tagNETDEVOutputSwitchesAlarmStatus  {  INT32 dwBooleanId;  INT32 dwChannelId;  INT32 enAlarmStatus;  } NETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S, \*LPNETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwBooleanId | 开关量编号 Boolean ID |
| dwChannelId | 通道ID,设备本身为0 |
| enAlarmStatus | 输出开关量报警状态 详情参见枚举[NETDEV\_RELAYOUTPUT\_STATE\_E](#_开关量状态枚举) |

### 计划（周）配置结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 布防计划是否使能,仅IPC支持使能开关 VMS、NVR默认使能; 详情参见枚举[NETDEV\_DEFENCE\_PLAN\_ENABLE\_E](#_布防计划使能枚举) |
| udwDayNum | 计划天数,NVR最大为8(一周七天和假日);IPC最大为7(一周七天) |
| astDayPlan | 一周内每天的布防计划列表 |
| byRes | 保留字段 |

### 计划（天）配置结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwIndex | 星期索引, 详情参见枚举[NETDEV\_LAPI\_WEEK\_INFO\_E](#_星期枚举) |
| udwSectionNum | 每天时间段个数 Section Num NVR最大为8段,IPC最大为4段 |
| astTimeSection | 布防时间段配置 |

### 时间段配置结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szBeginTime | 开始时间 |
| szEndTime | 结束时间 |
| udArmingType | 布防类型，详情参见枚举[NETDEV\_ARMING\_TYPE\_E](#_告警布防类型枚举) |

### 布防信息结构体

|  |
| --- |
| typedef struct tagNETDEVMonitorDefenceInfo  {  INT64 tBegin;  INT64 tEnd;  UINT32 udwTimeTemplateID;  BYTE byRes[128];  }NETDEV\_MONITOR\_DEFENCE\_INFO\_S, \*LPNETDEV\_MONITOR\_DEFENCE\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| tBegin | 时间模板生效起始时间(unix时间戳) |
| tEnd | 时间模板生效结束时间(unix时间戳) |
| udwTimeTemplateID | 时间模板索引，若未配置，则填写0 |
| byRes | 保留字段 |

### 添加布控返回的布控信息列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwChannelNum | 布控添加成功的实际通道数，需赋值标明内存申请大小 |
| pstMonitorChlInfos | 布控返回信息列表，内存由上层申请，不应小于下发的通道数量 |
| byRes | 保留字段 |

### 添加布控返回的布控信息结构体

|  |
| --- |
| typedef struct tagNETDEVMonitorChlInfo  {  UINT32 udwChannelID;  UINT32 udwResultCode;  UINT32 udwMonitorID;  }NETDEV\_MONITION\_CHL\_INFO\_S,\*LPNETDEV\_MONITION\_CHL\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwChannelID | 布控对应通道ID，IPC、VMS该字段不返回 |
| udwResultCode | 人脸处理结果码，详情参见枚举[NETDEV\_PERSON\_RESULT\_CODE\_E](#_人脸处理结果状态码枚举) |
| udwMonitorID | 相应通道对应的布控ID |

### 人员库信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 库ID |
| szName | 库名称 范围[1,63] |
| udwType | 人员库类型 详情参见枚举[NETDEV\_PEOPLE\_LIB\_TYPE\_E](#_人员库类型枚举) |
| udwPersonNum | 库中人员信息的总数 |
| udwFaceNum | 库中人脸照片总数 |
| udwMemberNum | 库中成员的总数 |
| udwLastChange | 库的信息的最后修改时间 |
| szBelongIndex | 库的唯一归属索引 |
| bIsMonitored | 是否已布控，获取库信息时必带 |
| byRes | 保留字段 |

### 人员库信息列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 设备中已创建的库数量 |
| pstLibInfo | 库信息,需动态分配内存 |
| byRes | 保留字段 |

### 删除库标志位结构体

|  |
| --- |
| typedef struct tagNETDEVDeleteDBFlagInfo  {  BOOL bIsDeleteMember;  UINT32 udwDevID;  BYTE byRes[124];  }NETDEV\_DELETE\_DB\_FLAG\_INFO\_S,\*LPNETDEV\_DELETE\_DB\_FLAG\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bIsDeleteMember | 是否删除库下里面的成员信息：0:不删除 1:删除 |
| udwDevID | 设备ID(仅VMS删除人脸库支持) |
| byRes | 保留字段 |

### 所有人员库的容量信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwMaxPerpleMun | 总库容量，单位：K人 |
| udwFreePerpleNum | 剩余容量，单位：人 |
| udwMaxLibNum | 最大可建库容量 |
| udwFreeLibNum | 剩余可建库容量 |
| udwNum | 已建库个数 库个数范围:[0, 16] |
| stLibCapInfoList | 单库容量信息 |
| byRes | 保留字段 |

### 人脸库容量信息结构体

|  |
| --- |
| typedef struct tagNETDEVPersonLibCapInfo  {  UINT32 udwLibID;  UINT32 udwCapacity;  BYTE byRes[128];  }NETDEV\_PERSON\_LIB\_CAP\_INFO\_S, \*LPNETDEV\_PERSON\_LIB\_CAP\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwLibID | 库ID |
| udwCapacity | 库容量信息，单位：人 |
| byRes | 保留字段 |

### 人员信息查询条件结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 查询条件数量 |
| pstQueryInfos | 查询条件列表，Num为0时，不带此字段 |
| udwLimit | 每次查询的数量，最大20 |
| udwOffset | 从当前序号开始查询，序号从0开始 |
| byRes | 保留字段 |

### 批量查询返回的基本信息结构体

|  |
| --- |
| typedef struct tagNETDEVBatchOperateBasicInfo  {  UINT32 udwTotal;  UINT32 udwOffset;  UINT32 udwNum;  BYTE byRes[128];  }NETDEV\_BATCH\_OPERATE\_BASIC\_S,\*LPNETDEV\_BATCH\_OPERATE\_BASIC\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTotal | 数量 |
| udwOffset | 查询起始序号 |
| udwNum | 查询结果总数 |
| byRes | 保留字段 |

### 删除人脸布控返回的信息列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 批量操作数量 |
| udwStatus | 结果状态 |
| pstBatchList | 批量操作信息 最大是2000个,需动态申请 |
| byRes | 保留字段 |

### 批量操作信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwReqSeq | 请求数据序号 |
| udwResultCode | 返回错误码,人脸布控操作结果详见  [NETDEV\_PERSON\_MONITOR\_OPT\_RES\_CODE\_E](#_人脸布控操作结果错误码枚举) |
| udwID | 编号 |
| szName | 成员名称，长度范围[1,63] |
| byRes | 保留字段 |

### 告警抓图URL查找条件

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道号 Channel number 或者 开关量告警编号，参数为通道号\*100+报警输出号，设备本身通道号为0 |
| tAlarmTime | 告警产生时间 |
| enAlarmType | 告警类型 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_FindAlarmSnapShotURL](#_获取告警联动抓图URL列表)

### 告警抓拍图片信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szURL | 图片URL |
| szName | 图片名称 |
| dwSize | 图片大小 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_FindNextAlarmSnapShotURL](#_逐个获取查找到的人员信息)

### 抓拍图片信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szURL | 图片URL |
| bSaveLocal | 是否保存在本地，TURE表示保存本地，FALSE表示写入缓存 |
| szFileName | 保存图片的本地路径及名称 |
| pcBuffer | 保存图片缓存,需客户申请内存，大小为dwSize缓存大小 |
| dwSize | 缓存大小 |
| dwCaptureMode | 保存图像格式，详见枚举 [NETDEV\_PICTURE\_FORMAT\_E](#_抓图图片格式枚举) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SaveSnapShotFile](#_保存告警抓图)

### 点坐标结构体

|  |
| --- |
| typedef struct tagNETDEVPointS  {  INT32 dwPointX;  INT32 dwPointY;  }NETDEV\_POINT\_S, \*LPNETDEV\_POINT\_S; |

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwPointX | 横坐标,万分比，取值范围[0-10000] |
| dwPointY | 纵坐标,万分比，取值范围[0-10000] |

**See also：**

[NETDEV\_SetMouseMoveMode](#_鼠标操作模式)、[NETDEV\_GetFishEyeParam](#_获取鱼眼矫正参数)

### 手动录像结构体

|  |
| --- |
| 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; |

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道号 |
| enRecordType | 录像类型，参考#[NETDEV\_RECORD\_TYPE\_E](#_录像类型枚举) |
| byRes | 预留字段 |

**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)

### 设备基本信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szDevModel | 设备型号 |
| szSerialNum | 硬件序列号 |
| szFirmwareVersion | 软件版本号 |
| szMacAddress | IPv4的Mac地址 |
| szDeviceName | 设备名称 |
| szManufacturer | 厂商信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### NTP参数

|  |
| --- |
| typedef struct tagNETDEVSystemNTPInfo  {  BOOL bSupportDHCP;  [NETDEV\_SYSTEM\_IPADDR\_S](#_地址) stAddr;  }NETDEV\_SYSTEM\_NTP\_INFO\_S, \*LPNETDEV\_SYSTEM\_NTP\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bSupportDHCP | 是否支持DHCP |
| stAddr | NTP 信息 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 地址结构体

|  |
| --- |
| typedef struct tagSysemIPAddr  {  INT32 eIPType;  CHAR szIPAddr[NETDEV\_LEN\_132];  }NETDEV\_SYSTEM\_IPADDR\_S, \*LPNETDEV\_SYSTEM\_IPADDR\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| eIPType | 协议类型参见枚举#[NETDEV\_HOSTTYPE\_E](#_IP协议类型) |
| szIPAddr | IP地址 |

**See also：**

NETDEV\_SYSTEM\_NTP\_INFO\_S

### NTP列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| ulNum | NTP服务器个数 |
| astNTPServerInfoList | NTP服务器列表，当前只支持一个NTP服务器 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 通道视频流信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| enStreamType | 码流索引 |
| bEnableFlag | 是否启用 |
| dwHeight | 视频编码分辨率-高 |
| dwWidth | 视频编码分辨率-宽 |
| dwFrameRate | 视频编码配置帧率 |
| dwBitRate | 码率 |
| enCodeType | 视频编码格式 |
| enQuality | 图像质量 |
| dwGop | I帧间隔 |
| bConstantBitRate | 是否为定码率0:变码率1:定码率 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 视频流信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 视频流个数 |
| astVideoStreamInfoList | 视频流信息列表 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 视频流信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 视频流是否启用编码 |
| udwStreamID | 码流索引，参见枚举[NETDEV\_LIVE\_STREAM\_INDEX\_E](#_码流类型枚举) |
| udwMainStreamType | 主码流类型，参见[NETDEV\_MAIN\_STREAM\_TYPE\_E](#_主码流类型) |
| stVideoEncodeInfo | 视频编码参数信息 |

**See also：**

[NETDEV\_VIDEO\_STREAM\_INFO\_LIST\_S](#_视频流信息列表)

### 视频编码参数信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnableSVCMode | SVC配置,0：关闭,1：开启 |
| udwEncodeFormat | 视频编码格式信息，参见枚举[NETDEV\_VIDEO\_CODE\_TYPE\_E](#_视频编码格式枚举) |
| udwWidth | 图像宽度 |
| udwHeight | 图像高度 |
| udwBitrate | 码率 |
| udwBitrateType | 码率类型，参见[NETDEV\_BIT\_RATE\_TYPE\_E](#_码率模式类型) |
| udwFrameRate | 帧率 |
| udwGopType | Gop模式,参见[NETDEV\_GOP\_TYPE\_E](#_GOP类型) |
| udwIFrameInterval | I帧间隔，范围根据能力来定 |
| udwImageQuality | 图像质量，范围[1, 9]，9代表图像质量最高 |
| udwSmoothLevel | 码流平滑等级，范围[1,9]，1代表平滑级别最低 |
| udwSmartEncodeMode | 智能编码模式，参见[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)

### 视频制式能力

|  |
| --- |
| typedef struct tagNETDEVVideoModeInfo  {  UINT32 udwWidth;  UINT32 udwHeight;  UINT32 udwFrameRate;  }NETDEV\_VIDEO\_MODE\_INFO\_S,\*LPNETDEV\_VIDEO\_MODE\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwWidth | 图像宽度 |
| udwHeight | 图像高度 |
| udwFrameRate | 图像帧率 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 通道OSD的基本属性信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stTimeOSD | 通道的时间OSD信息 |
| stNameOSD | 通道的名称OSD信息 |
| wTextNum | 字符OSD个数 |
| astTextOverlay | 通道OSD字符叠加信息 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### OSD时间配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnableFlag | 时间OSD使能, TRUE为使能,FALSE为非使能 |
| bWeekEnableFlag | 是否显示星期(预留) |
| stAreaScope | 区域坐标 |
| udwTimeFormat | 时间OSD时间格式,详见[NETDEV\_OSD\_TIME\_FORMAT\_CAP\_E](#_OSD时间格式能力集枚举) |
| udwDateFormat | 日期OSD时间格式,详见[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字符叠加信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnableFlag | OSD字符叠加使能, TRUE为使能,FALSE为非使能 |
| stAreaScope | OSD字符叠加区域坐标 |
| szOSDText | OSD字符叠加名称字符串 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_VIDEO\_OSD\_CFG\_S](#_通道OSD的基本属性信息)

### 区域

|  |
| --- |
| typedef struct tagNETDEVAreaScope  {  INT32 dwLocateX;  INT32 dwLocateY;  }NETDEV\_AREA\_SCOPE\_S, \*LPNETDEV\_AREA\_SCOPE\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwLocateX | 顶点x坐标值[0,10000] |
| dwLocateY | 顶点y坐标值[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)

### 通道OSD配置列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | OSD个数 |
| stChlOsdCfg | OSD配置 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 通道OSD配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwOsdID | OSD ID |
| enOsdTextType | OSD类型 |
| stAreaScope | OSD坐标 |
| szTextInfo | OSD内容 |

**See also：**

[NETDEV\_CHL\_OSD\_CFG\_LIST\_S](#_通道OSD配置列表)

### 通道OSD内容样式

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwFontStyle | 字体形式，参见枚举[NETDEV\_OSD\_FONT\_STYLE\_E](#_OSD字体形式枚举) |
| udwFontSize | 字体大小，参见枚举[NETDEV\_OSD\_FONT\_SIZE\_E](#_OSD字体大小枚举) |
| udwColor | 颜色 |
| udwDateFormat | 日期格式，参见枚举[NETDEV\_OSD\_DATE\_FORMAT\_E](#_日期格式) |
| udwTimeFormat | 时间格式，参见枚举[NETDEV\_OSD\_TIME\_FORMAT\_E](#_OSD时间格式枚举) |
| audwFontAlignList | 区域内字体对齐，固定个区域，IPC支持,参见枚举[NETDEV\_OSD\_ALIGN\_E](#_OSD区域内对齐枚举) |
| udwMargin | 边缘空的字符数，IPC支持，参见枚举[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)

### 所有告警开关量输入信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 开关量数量 |
| astAlarmOutputInfo | 开关量配置信息 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 告警开关量输出信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szName | 开关量名称 |
| dwChancelId | 通道号 |
| enDefaultStatus | 输出开关量默认状态参见#[NETDEV\_BOOLEAN\_MODE\_E](#_输入开关量运行模式) |
| dwDurationSec | 告警持续时间单位S |
| dwOutputNum | 告警输出序号 |

**See also**：[NETDEV\_ALARM\_OUTPUT\_LIST\_S](#_所有告警开关量输入信息)

### 触发开关量输入告警

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szName | 开关量名称 |
| enOutputState | 触发状态 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 所有告警开关量输入信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 输入开关量数量 |
| astAlarmInputInfo | 输入开关量配置信息 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 告警开关量输入信息

|  |
| --- |
| typedef struct tagNETDEVAlarmInputInfo  {  CHAR szName[NETDEV\_LEN\_64];  }NETDEV\_ALARM\_INPUT\_INFO\_S, \*LPNETDEV\_ALARM\_INPUT\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szName | 输入开关量名称 |

**See also**：

[NETDEV\_ALARM\_INPUT\_LIST\_S](#_所有告警开关量输入信息_1)

### 输出开关量的逻辑报警状态列表(手动告警)

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 设备报警输出开关量通道数 |
| astOutputAlarmStatusInfo | 输出开关量报警状态列表 |
| bRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 输出开关量的逻辑报警状态(手动告警)

|  |
| --- |
| typedef struct tagNETDEVOutputSwitchesAlarmStatus  {  INT32 dwBooleanId;  INT32 dwChannelId;  INT32 enAlarmStatus;  }NETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S, \*LPNETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwBooleanId | 开关量编号 |
| dwChannelId | 通道ID,设备本身为0 |
| enAlarmStatus | 输出开关量报警状态参见#[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))

### 触发或清除输出开关量的逻辑报警状态结构体(手动告警)

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 需要设置的报警输出开关量数量 |
| dwIDList | 需要设置的输出开关量编号列表 |
| dwAlarmAction | 执行的命令 |
| bRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 输入开关量告警配置信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 输入开关量告警数量 |
| astInputSwitchesInfo | 输入开关量告警配置信息 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 输入开关量告警信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChancelId | 通道号 |
| dwInputSwitchIndex | 开关量索引 |
| szName | 输入开关量名称 |
| szGBID | 国标资源编码,仅IPC支持 |
| dwRunMode | 报警类型1:常开2：常闭 |
| dwEnabled | 报警使能0:使能1：使能 |
| dwVideoResID | 对应的视频通道ID,仅VMS支持 |
| bRes | 保留字段 |

**See also**：

[NETDEV\_INPUT\_SWITCH\_INFO\_LIST\_S](#_输入开关量告警配置信息列表)

### 设备图像设置

|  |
| --- |
| typedef struct tagNETDEVImagingSetting  {  INT32 dwContrast; /\* 对比度 \*/  INT32 dwBrightness; /\* 亮度 \*/  INT32 dwSaturation; /\* 饱和度 \*/  INT32 dwSharpness; /\* 亮度 \*/  BYTE byRes[252]; /\* 保留字段 \*/  }NETDEV\_IMAGE\_SETTING\_S, \*LPNETDEV\_IMAGE\_SETTING\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwContrast | 对比度 |
| dwBrightness | 亮度 |
| dwSaturation | 饱和度 |
| dwSharpness | 亮度 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 图像曝光参数

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwMode | 曝光模式枚举详见#[NETDEV\_EXPOSURE\_MODE\_E](#_曝光模式) |
| dwCompensationLevel | 曝光补偿级别,曝光模式为非手动曝光模式时可用。范围[-100,100].图像能力集支持该功能，此字段必选 |
| udwHLCSensitivity | 强光抑制灵敏度，当前场景为道路强光抑制及园区强光抑制时可用,范围[1,9]。图像能力集支持该功能，此字段必选 |
| stIrisInfo | 光圈信息。图像能力集支持该功能，此字段必选 |
| stShutterInfo | 快门信息。图像能力集支持该功能，此字段必选 |
| stGainInfo | 增益信息 |
| stWideDynamicInfo | 宽动态信息。图像能力集支持该功能，此字段必选 |
| stMeteringInfo | 测光信息。当前场景不是道路强光抑制及园区强光抑制时可用。图像能力集支持该功能，此字段必选 |
| stDayNightInfo | 昼夜模式信息。图像能力集支持该功能，此字段必选 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 光圈信息

|  |
| --- |
| typedef struct tagNETDEVIrisInfo  {  UINT32 udwIris;  UINT32 udwMinIris;  UINT32 udwMaxIris;  BYTE byRes[128];  }NETDEV\_IRIS\_INFO\_S, \*LPNETDEV\_IRIS\_INFO\_S |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwIris | 光圈,在光圈优先、手动曝光模式下可选。光圈支持的取值:160，200，240，280，340，400，480，560，680，800，960，1100，, 1600, 2200 |
| udwMinIris | 最小光圈值自定义曝光模式下可用，枚举同Iris能力集所描述，不得大于光圈最大值。图像能力集支持该功能，此字段必选 |
| udwMaxIris | 最大光圈值自定义曝光模式下可用，枚举同Iris能力集所描述，不得小于光圈最小值。图像能力集支持该功能，此字段必选 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)

### 快门信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwShutterTime | 快门时间枚举见#[NETDEV\_SHUTTER\_TIME\_RANGE\_E](#_快门时间支持的取值),快门时间单位 0：微秒1：秒 |
| udwMinShutterTime | 快门时间最小值枚举见#[NETDEV\_SHUTTER\_TIME\_RANGE\_E](#_快门时间支持的取值) |
| udwMaxShutterTime | 快门时间最大值枚举见#[NETDEV\_SHUTTER\_TIME\_RANGE\_E](#_快门时间支持的取值) |
| udwIsEnableSlowShutter | 慢快门使能。非光圈优先模式下可用：0：不使能 1：使能 |
| udwSlowestShutter | 最慢慢快门,慢快门使能后可用 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)

### 增益信息

|  |
| --- |
| typedef struct tagNETDEVGainInfo  {  UINT32 udwGain;  UINT32 udwMinGain;  UINT32 udwMaxGain;  BYTE byRes[128];  }NETDEV\_GAIN\_INFO\_S, \*LPNETDEV\_GAIN\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwGain | 增益值（单位:db）手动曝光模式下可用。范围[1,100] |
| udwMinGain | 增益最小值,自定义曝光模式下可用，不得大于增益最大值。最小值为1 |
| udwMaxGain | 增益最大值, 自定义曝光模式下可用，不得小于增益最小值。最大值为100 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)

### 宽动态信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwWideDynamicMode | 宽动态模式 枚举详见#[NETDEV\_WIDE\_DYNAMIC\_MODE\_E](#_测光控制模式) |
| udwWideDynamicLevel | 宽动态级别配置，宽动态开启且在曝光模式为自动模式、自定义、快门优先、室内HZ、室内HZ、低拖影下可用。范围[1, 9] |
| udwOpenSensitivity | 宽动态开启的灵敏度。宽动态模式为自动下可用。范围[1, 9] |
| udwCloseSensitivity | 宽动态关闭的灵敏度。宽动态模式为自动下可用。范围[1, 9] |
| udwAntiFlicker | 宽动态条纹抑制：：关闭1：开启该功能开启后，可消除图像中的条纹效应 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)

### 测光信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwMeteringMode | 测光控制模式,此字段在非手动曝光模式下可用。枚举详见#[NETDEV\_DAY\_NIGHT\_MODE\_E](#_昼夜模式类型) |
| udwRefBrightness | 人脸亮度。人脸测光模式下可用。范围：[0, 100] |
| udwHoldTime | 最短持续时间。人脸测光模式下可用。单位：分钟。范围：[0, 60] |
| stMeteringArea | 测光区域,在测光模式为区域测光及点测光时，此字段可用 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)

### 测光区域

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stAreaTopLeft | 左上角区域 |
| stAreaBotRight | 右下角区域 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_METERING\_INFO\_S](#_测光信息_1)

### 左上角区域

|  |
| --- |
| typedef struct tagNETDEVAreaTopLeft  {  UINT32 dwTopLeftX;  UINT32 dwTopLeftY;  BYTE byRes[128];  }NETDEV\_AREA\_TOP\_LEFT\_S, \*LPNETDEV\_AREA\_TOP\_LEFT\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwTopLeftX | 左上角横坐标(比例)：区域测光模式范围: [0, 100] |
| dwTopLeftY | 左上角纵坐标(比例)：区域测光模式范围: [0, 100] |
| byRes | 保留字段 |

**See al**so：

[NETDEV\_METERING\_AREA\_S](#_测光区域)

### 右下角区域

|  |
| --- |
| typedef struct tagNETDEVAreaBotRight  {  UINT32 dwBottomRightX;  UINT32 dwBottomRightY;  BYTE byRes[128];  }NETDEV\_AREA\_BOT\_RIGHT\_S, \*LPNETDEV\_AREA\_BOT\_RIGHT\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwBottomRightX | 左上角横坐标(比例)：区域测光模式范围: [0, 100] |
| dwBottomRightY | 左上角纵坐标(比例)：区域测光模式范围: [0, 100] |
| byRes | 保留字段 |

**See also**：

[NETDEV\_METERING\_AREA\_S](#_测光区域)

### 昼夜模式信息

|  |
| --- |
| typedef struct tagNETDEVDayNightInfo  {  UINT32 udwDayNightMode;  UINT32 udwDayNightSensitivity;  UINT32 udwDayNightTime;  BYTE byRes[128];  }NETDEV\_DAY\_NIGHT\_INFO\_S, \*LPNETDEV\_DAY\_NIGHT\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwDayNightMode | 昼夜模式类型 枚举参见#[NETDEV\_DAY\_NIGHT\_MODE\_E](#_昼夜模式类型) |
| udwDayNightSensitivity | 昼夜模式灵敏度DayNightSensitivity 在昼夜模式为自动模式下可用，范围[0, 9]。若图像能力支持该功能，此字段必选 |
| udwDayNightTime | 昼夜模式切换时间，在昼夜模式为自动模式下可用。范围[3, 120]。单位秒。若图像能力支持该功能，此字段必选 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)

### 补光灯信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwEnabled | 补光灯使能开关：：关闭1：开启 |
| udwType | 补光灯类型详见枚举类型#[NETDEV\_LAMP\_TYPE\_E](#_补光灯类型) |
| udwMode | 补光灯控制模式详见枚举类型#[NETDEV\_LAMP\_CTRL\_MODE\_E](#_补光灯支持的控制模式) |
| udwNearLevel | 近光灯级别,在红外控制模式为手动控制及手动-强制开启模式下,此字段可用。补光灯能力集支持该功能，此字段必选。范围[0-1000]。级别越高，补光灯强度越大 |
| udwMiddleLevel | 中光灯级别,在红外控制模式为手动控制及手动-强制开启模式下可用。补光灯能力集支持该功能，此字段必选。范围[0-1000]。级别越高，补光灯强度越大 |
| udwFarLevel | 远光灯级别,在红外控制模式为手动控制及手动-强制开启模式下可用。补光灯能力集支持该功能，此字段必选。范围[0-1000]。级别越高，补光灯强度越大 |
| udwSuperFarLevel | 超远光灯级别,在红外控制模式为手动控制及手动-强制开启模式下可用。补光灯能力集支持该功能，此字段必选。范围[0-1000]。级别越高，补光灯强度越大 |
| udwLaserAngle | 激光角度，当补光灯类型为激光器且，在红外控制模式为手动控制及手动-强制开启模式下可用。补光灯能力集支持该功能，此字段必选。范围见补光灯能力集。激光角度越小能量越集中，角度越大能量越分散 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 白平衡信息

|  |
| --- |
| typedef struct tagNETDEVWhiteBalanceInfo  {  UINT32 udwMode;  INT32 dwRedOffset;  INT32 dwBlueOffset;  BYTE byRes[256];  }NETDEV\_WHITE\_BALANCE\_INFO\_S, \*LPNETDEV\_WHITE\_BALANCE\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwMode | 白平衡模式详见枚举类型#[NETDEV\_WHITE\_BALANCE\_MODE\_E](#_白平衡模式) |
| dwRedOffset | Red偏移值，只有在Mode为[1,5,8]微调模式下可配置 |
| dwBlueOffset | Blue偏移值，只有在Mode为[1,5,8]微调模式下可配置 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 网络配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwMTU | MTU值 |
| bIPv4DHCP | IPv4的DHCP |
| szIpv4Address | IPv4的IP地址 |
| szIPv4GateWay | IPv4的网关地址 |
| szIPv4SubnetMask | IPv4的子网掩码 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### VPN客户端指定模式信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwEnabled | 是否开启对应兼容格式的一个开关: 关闭1: 开启 |
| udwCompatibilityMode | 对应所设置的兼容模式0: Unp1.0（unp）1: Unp2.0 (softvpn) |
| udwAuthEnabled | 鉴权是否开启。若为Unp2.0一定鉴权，为开启状态。0：关闭1：开启 |
| udwEncryptEnabled | 加密是否开启兼容模式, 若当前开启兼容模式为Unp1.0 此字段需为使能关闭。0：关闭1：开启 |
| udwInterface | 可选IPC 需要；NVR和VMS不需要。有线网口：,4G 网口：。范围：[0,1] |
| szLoginName | 可选：AuthEnable开启时，带此字段。服务器的用户名，不支持中文名。范围[0-63] |
| szPIN | 可选：AuthEnable开启时，带此字段服务器的用户密码范围[0,256],不修改接入登录密码时不必携带，修改接入登录密码时必携带 |
| udwAddressType | AddressType：IP地址类型。0: IPv4 1: IPv6 2:域名3: IPv4和IPv6都需要.当前仅支持IPv4 |
| szAddress | 地址 |
| udwPort | 可选;NVR 和VMS需要此字段，对应兼容格式的服务器端口，范围[1-65535]，兼容Unp1.0默认值1701；兼容Unp2.0 默认是5555 |
| udwUNPSuccessd | UNP成功连接标识符(查询返回表示当前对应UNP有无分配对应ip,掩码,路由)。成功，失败，获取时携带 |
| szUNPShareIPAddr | UNP成功连接服务器所分配的对应地址(查询并连接成功才返回,非输入参数) |
| szUNPShareMask | UNP成功连接服务器所分配的对应掩码(查并连接成功才返回,非输入参数) |
| szUNPShareWayAddr | UNP成功连接服务器所分配的对应的路由(查询并连接成功才返回,非输入参数) |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 设备DNS信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | DNS地址数量 |
| astDNSList | DNS地址列表，第个为首要地址，第个为次选地址 |
| byRes | 保留字 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### DNS地址

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwAddressType | IP地址类型0: IPv4 1: IPv6 3: IPv4和IPv6都需要当前仅支持IPv4 |
| szIPAddress | IPv4地址 |
| szIPv6Address | IPv6地址 |
| byRes | 保留字 |

**See also：**

[NETDEV\_DNS\_INFO\_S](#_设备DNS信息)

### 网卡信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 网卡个数 |
| udwDefaultRouteNIC | 作默认路由的网卡ID，当存在多张网卡（不含内部网卡）时必选 |
| udwWorkMode | 网卡工作模式0：多址设定1：负载均衡 2：网络容错 |
| astNetworkInterfaceList | 网卡信息列表，当Num值为时该节点可选 |
| byRes | 保留字 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 网卡的详细信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 网卡编号对VMS、IPC从开始对NVR该字段是资源编码 |
| szName | 网卡名称，NVR、IPC支持 |
| udwWorkMode | 网卡工作模式0：多址设定1：负载均衡 2：网络容错 |
| bIsInnerNIC | 是否为内部网卡 |
| szInnerNICIPAddress | 内部网卡IP地址 |
| szInnerNICNetmask | 内部网卡网络掩码；该字段不可配置，仅供查询 |
| szInnerNICName | 内部网卡名称；该字段不可配置，仅供查询 |
| udwMTU | MTU长度，范围[576,1500] |
| szMAC | MAC地址，只读属性，字符串长度范围[0, 48] |
| udwNegotiationMode | 网口工作模式 |
| stIPV4Infos | IPv4信息，如果不支持IPv4，则该节点可选 |
| stIPV6Infos | IPv6信息，如果不支持IPv6，则该节点可选 |
| byRes | 保留字 |

**See also：**

[NETDEV\_NETWORK\_CARD\_INFO\_S](#_网卡信息)

### 网卡信息中IPV4信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwIPGetType | IP获取方式0：for static 1：for PPPOE2：for DHCP |
| szLoginName | PPPoE账号 |
| szPIN | PPPOE密码（MD5加密） |
| udwAddressNum | IP地址个数 |
| astIPv4AddressInfo | IPv4信息，如果不支持IPv4，则该节点可选 |

**See also：**

[NETDEV\_NETWORK\_INTERFACE\_INFO\_S](#_网卡的详细信息)

### IPV4详细地址信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szAddress | IP地址 |
| szNetmask | 子网掩码 |
| szGateway | 默认网关 |

**See also：**

[NETDEV\_NETWORK\_IPV4\_INFO\_S](#_网卡信息中IPV4信息)

### 网卡信息中IPV6信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwIPGetType | IP获取方式 |
| udwAddressNum | IP地址个数 |
| astIPv6AddressInfo | IPv6信息，如果不支持IPv6，则该节点可选 |

**See also：**

[NETDEV\_NETWORK\_INTERFACE\_INFO\_S](#_网卡的详细信息)

### IPV6详细地址信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPrefixLenth | 子网前缀长度，范围为[3,127] |
| szAddress | IP地址 |
| szGateway | 默认网关 |

**See also：**

[NETDEV\_NETWORK\_IPV6\_INFO\_S](#_网卡信息中IPV6信息)

### 端口信息

|  |
| --- |
| typedef struct tagNETDEVNetworkPortsInfo  {  UINT32 udwHttpPort;  UINT32 udwHttpsPort;  UINT32 udwRtspPort;  BYTE byRes[64];  }NETDEV\_NETWORK\_PORTS\_INFO\_S, \*LPNETDEV\_NETWORK\_PORTS\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwHttpPort | Http端口 |
| udwHttpsPort | Https端口 |
| udwRtspPort | Rtsp端口 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 隐私遮盖配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 区域个数 |
| astArea | 遮盖区域参数 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 区域配置结构体定义

|  |
| --- |
| typedef struct tagAreaInfo  {  INT32 bIsEanbled;  INT32 dwTopLeftX;  INT32 dwTopLeftY;  INT32 dwBottomRightX;  INT32 dwBottomRightY;  INT32 dwIndex;  }NETDEV\_PRIVACY\_MASK\_AREA\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bIsEanbled | 是否启用 |
| dwTopLeftX | 左上角X [0, 10000] |
| dwTopLeftY | 左上角Y [0, 10000] |
| dwBottomRightX | 右下角X [0, 10000] |
| dwBottomRightY | 右下角Y [0, 10000] |
| dwIndex | 索引 |

**See also：**

[NETDEV\_PRIVACY\_MASK\_CFG\_S](#_隐私遮盖配置信息)

### 遮挡检测分析信息

|  |
| --- |
| typedef struct tagNETDEVTamperAlarmInfo  {  INT32 dwSensitivity;  BYTE byRes[256];  }NETDEV\_TAMPER\_ALARM\_INFO\_S, \*LPNETDEV\_TAMPER\_ALARM\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSensitivity | 灵敏度 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 运动检测分析信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSensitivity | 灵敏度 |
| dwObjectSize | 物体大小 |
| dwHistory | 持续时间 |
| awScreenInfo | 屏幕宏块信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 越界检测分析信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnable | 是否启用 |
| stAreaInfo | 区域信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 区域信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnable | 是否启用 |
| dwSensitivity | 灵敏度 |
| dwDirection | 触发方向 |
| dwTopLeftX | 左上角X [0, 10000] |
| dwTopLeftY | 左上角Y [0, 10000] |
| dwBottomRightX | 右下角X [0, 10000] |
| dwBottomRightY | 右下角Y [0, 10000] |

**See also：**

[NETDEV\_CROSS\_LINE\_ALARM\_INFO\_S](#_越界检测分析信息)

### 入侵检测分析信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnable | 是否启用 |
| stAreaInfo | 区域信息，最多支持设置个区域，每个区域由三到六边形构成 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 入侵检测区域信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnable | 是否启用 |
| dwSensitivity | 灵敏度，范围1-100 |
| dwTimeThreshold | 时间阈值范围1-10 |
| percentage | 比率范围1-100 |
| dwPointCount | 区域点个数，范围3-6，如果不设置该区域，点个数将忽略 |
| stPointInfo | 区域点坐标信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_INTRUSION\_ALARM\_INFO\_S](#_入侵检测分析信息)

### 入侵检测区域点坐标

|  |
| --- |
| typedef struct tagNETDEVIntrusionPointInfo  {  INT32 dwPointX;  INT32 dwPointY;  }NETDEV\_INTRUSION\_POINT\_INFO\_S, \*LPNETDEV\_INTRUSION\_POINT\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwPointX | X [0, 10000] |
| dwPointY | Y [0, 10000] |

**See also：**

[NETDEV\_INTRUSION\_AREA\_INFO\_S](#_入侵检测区域信息)

### 硬盘信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 硬盘个数 |
| astDisksInfo | 硬盘信息 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 硬盘详细信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwDiskCabinetIndex | 硬盘索引 |
| dwSlotIndex | 硬盘所在槽位索引 |
| dwTotalCapacity | 硬盘总容量 |
| dwUsedCapacity | 已经使用量 |
| enStatus | 运行状态 |
| szManufacturer | 厂商 |

**See also：**

[NETDEV\_DISK\_INFO\_LIST\_S](#_硬盘信息列表)

### 聚焦信息

|  |
| --- |
| typedef struct tagNETDEVFocusInfo  {  [NETDEV\_FOCUS\_MODE\_E](#_聚焦模式) enFocusMode;  BYTE bRes[16];  }NETDEV\_FOCUS\_INFO\_S, \*LPNETDEV\_FOCUS\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| enFocusMode | 聚焦模式 |
| bRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 昼夜模式信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| enIrCutFilterMode | 昼夜模式 |
| bRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 透雾信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| enDefoggingMode | 除雾模式 |
| fDefoggingLevel | 除雾等级(0.0, 1.0) |
| bRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 指定通道的图像增强参数

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwBrightness | 亮度 |
| udwContrast | 对比度 |
| udwSaturation | 饱和度 |
| udwSharpness | 锐度 |
| udwImageRotation | 图像镜像模式 |
| udw2DNoiseReduce | 2D降噪 |
| udw3DNoiseReduce | 3D降噪 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 音频输入参数配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bIsMute | 是否静音0：不静音，静音 |
| udwType | 采集口类型参见枚举#[NETDEV\_AUDIO\_COLLECTION\_PORT\_TYPE\_E](#_音频采集口枚举) |
| udwEncodeFormat | 音频编码格式参见枚举#[NETDEV\_AUDIO\_ENCODE\_FORMAT\_TYPE\_E](#_音频编码格式类型枚举) |
| udwSampleRate | 音频采样率参见枚举#[NETDEV\_AUDIO\_SAMPLING\_RATE\_E](#_音频采样率枚举) |
| udwInputGain | 音频增益，范围[0,255] |
| bNoiseReductionEnabled | 是否启用降噪0不启用，启用 |
| dwAudioInNum | 音频输入数量 |
| astAudioInputInfoList | 音频输入采集信息列表，当采集口类型为  [NETDEV\_AUDIO\_COLLECTION\_PORT\_TYPE\_AUDIO\_IN](#_音频采集口枚举)时有效 |
| dwSerialInNum | 串口输入数量 |
| astSerialInputInfoList | 串口音频采集信息列表 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 音频输入通道信息

|  |
| --- |
| typedef struct tagNETDEVAudioInputInfo  {  INT32 dwChannelID;  UINT32 udwEnabled;  UINT32 udwMode;  BYTE byRes[128];  }NETDEV\_AUDIO\_INPUT\_INFO\_S, \*LPNETDEV\_AUDIO\_INPUT\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 音频通道号 |
| udwEnabled | 音频输入通道是否启动 |
| udwMode | 音频输入通道模式参见枚举#[NETDEV\_AUDIO\_CHL\_MODE\_E](#_音频输入模式枚举) |
| byRes | 保留字节 |

**See also：**

[NETDEV\_AUDIO\_INPUT\_CFG\_INFO\_S](#_音频输入参数配置信息)

### 夏令时配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnableDST | 夏令时使能 |
| stTimeDSTCfg | 夏令时配置 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 夏令时配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stBeginTime | 夏令时开始时间 |
| stEndTime | 夏令时结束时间 |
| dwOffsetTime | 夏令时节约时间参见枚举# [NETDEV\_DST\_OFFSET\_TIME](#_夏令时偏移时间) |

**See also：**

[NETDEV\_DST\_CFG\_S](#_夏令时配置信息)

### 时间配置

|  |
| --- |
| typedef struct tagNETDEVSystemTimeDST  {  INT32 dwMonth;  INT32 dwWeekInMonth;  INT32 dwDayInWeek;  INT32 dwHour;  }NETDEV\_TIME\_DST\_S, \*LPNETDEV\_TIME\_DST\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwMonth | 月(1~12) |
| dwWeekInMonth | 每月的第N周（1~5） |
| dwDayInWeek | 每周的星期几参见#[NETDEV\_DAY\_IN\_WEEK\_E](#_星期) |
| dwHour | 时 |

**See also：**

[NETDEV\_TIME\_DST\_CFG\_S](#_时间配置)

### 录像计划配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bPlanEnable | 计划使能 |
| bRedundantStorage | 冗余录像使能仅NVR支持 |
| stRecordRule | 录像计划规则仅NVR支持 |
| stWeekPlan | 计划配置仅NVR支持 |
| udwChlID | 视频输入通道号批量获取/添加时使用 |
| udwReqSeq | 请求数据序号[1, 50] 仅VMS支持添加录像计划Post必选 |
| udwTamplateID | 时间模板ID |
| szTamplateName | 时间模板名称仅VMS支持Get接口返回 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 录像计划规则

|  |
| --- |
| typedef struct tagNETDEVRecordRule  {  UINT32 udwPreRecordTime;  UINT32 udwPostRecordTime;  BYTE byRes[32];  }NETDEV\_RECORD\_RULE\_S, \*LPNETDEV\_RECORD\_RULE\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPreRecordTime | 警前预录时间,参考[NETDEV\_PRE\_RECORD\_TIME\_E](#_警前预录时间枚举) |
| udwPostRecordTime | 参考[NETDEV\_POST\_RECORD\_TIME\_E](#_警后预录时间枚举定义) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_RECORD\_PLAN\_CFG\_INFO\_S](#_录像计划配置信息)

### 视频计划（周）配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 布防计划是否使能,仅IPC支持使能开关VMS NVR默认使能;参考[NETDEV\_DEFENCE\_PLAN\_ENABLE\_E](#_布防计划使能枚举) |
| udwDayNum | 计划天数,NVR最大为(一周七天和假日);IPC最大为(一周七天) |
| astDayPlan | 一周内每天的布防计划列表 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_RECORD\_PLAN\_CFG\_INFO\_S](#_录像计划配置信息)

### 视频计划（天）配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwIndex | 星期索引,参见：[NETDEV\_LAPI\_WEEK\_INFO\_E](#_星期枚举) |
| udwSectionNum | 每天时间段个数 Section Num NVR最大为段,IPC最大为段 |
| astTimeSection | 布防时间段配置 |

**See also：**

[NETDEV\_VIDEO\_WEEK\_PLAN\_S](#_视频计划（周）配置)

### 视频时间段配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szBeginTime | 开始时间，如"00:00:00" |
| szEndTime | 结束时间，如"24:00:00" |
| udArmingType | 布防类型参考[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)

### 视频输入通道的抓图信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bIsEnabled | 抓图使能,0: 关闭, 1: 开启 |
| udwPictureMaxSize | 最大图片大小 |
| udwSnapshotInterval | 抓图间隔，范围[1, 60],单位:秒 |
| udwSnapshotNum | 抓图数量，范围[1,3] |
| stResolution | 抓图分辨率 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 视频源分辨率信息

|  |
| --- |
| typedef struct tagNETDEVVideoResolution  {  INT32 dwWidth;  INT32 dwHeight;  }NETDEV\_VIDEO\_RESOLUTION\_S, \*LPNETDEV\_VIDEO\_RESOLUTION\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwWidth | 宽度 |
| dwHeight | 高度 |

**See also：**

[NETDEV\_VIDEO\_SNAPSHOT\_S](#_视频输入通道的抓图信息)

### 计划（周）配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 使能,仅IPC支持 |
| udwNum | 计划天数，NVR最大为(一周七天和假日)IPC最大为(一周七天) |
| astDayPlanInfo | 一周内每天的布防计划列表 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 计划（天）配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 星期索引1：周一;2：周二；3：周三；4：周四；5：周五；6：周六；7：周日；8：假日 |
| udwNum | 每天时间段个数NVR最大为段；IPC最大为段 |
| astTimeSection | 时间段配置同一天各段时间不允许有重合 |

**See also：**

[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置)

### 时间段配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szBeginTime | 开始时间，如"00:00:00" |
| szEndTime | 结束时间，如"24:00:00" |
| udwArmingType | 布防类型: 0定时，1: 动检，2: 报警，3: 动检和报警，4: 动检或报警，5: 无计划,10: 事件 |

**See also：**

[NETDEV\_DAY\_PLAN\_INFO\_S](#_计划（天）配置)

### 开关量计划（周）配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSwitchIndex | 开关量索引 |
| astWeekPlanInfo | 周计划配置信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 录像状态信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSize | 录像状态数量 |
| astRecordStatus | 录像状态信息 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 录像状态信息

|  |
| --- |
| typedef struct tagNETDEVRecordStatus  {  INT32 dwChannelID;  INT32 dwRecordType;  INT32 dwRecordStatus;  BYTE byRes[128];  }NETDEV\_RECORD\_STATUS, \*LPNETDEV\_RECORD\_STATUS; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道号 |
| dwRecordType | 录像类型0:手动录像，1:事件录像，2:交易录像，3:定时录像，4:其他 |
| dwRecordStatus | 录像状态0:正在录像，1:未启动存储，2:没有硬盘或硬盘坏，3:通道不在线 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_RECORD\_STATUS\_LIST\_S](#_录像状态信息列表)

### 串口信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSerialPortNum | 串口数 |
| pstSerialPortList | 各串口配置列表当dwSerialPortNum为0时该字段可,dwSerialPortNum非0时，需动态分配内存 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 串口配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 串口号 |
| dwMode | 串口工作模式1：云台控制，2：透明通道 |
| stSerialParam | 串口参数 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SERIAL\_PORTS\_S](#_串口信息)

### 串口参数

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwBaudRate | 串口波特率。范围中的特殊值，范围：[1200,115200]，单位: bps |
| dwDataBit | 数据位，范围[5-8] |
| dwFlowCtrl | 流控制0：无，1：软流控，2：硬流控 |
| dwParity | 校验位0：无校验，1：奇校验，2：偶校验 |
| dwStopBit | 停止位 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SERIAL\_PORT\_S](#_串口配置信息)

### 随路音频状态信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 数量 |
| astAudioDecStatus | 随路音频状态信息 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 随路音频状态信息

|  |
| --- |
| typedef struct tagNETDEVAudioDecodeStatus  {  INT32 dwStreamID;  BOOL bDecodeAudio;  }NETDEV\_AUDIO\_DECODE\_STATUS\_S, \*LPNETDEV\_AUDIO\_DECODE\_STATUS\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwStreamID | 视频流索引号 |
| bDecodeAudio | 是否开启随路音频解码 |

**See also：**

[NETDEV\_AUDIO\_DECODE\_STATUS\_LIST\_S](#_随路音频状态信息列表)

### 视频丢失告警的配置信息

|  |
| --- |
| typedef struct tagNETDEVVideoLossRuleInfo  {  BOOL bEnabled;  BYTE byRes[256];  }NETDEV\_VIDEO\_LOSS\_RULE\_INFO\_S, \*LPNETDEV\_VIDEO\_LOSS\_RULE\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 使能, 0:不使能1:使能 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 布控任务联动动作列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 布控任务联动动作个数 |
| stActionInfo | 布控任务联动动作 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 布控任务联动动作

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwActID | 联动动作ID，参见枚举值[NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举) |
| stEnabledInfo | 联动参数使能标记，适用于联动蜂鸣器、联动EMail、联动告警弹窗 |
| stOutputSwitchActParamInfo | 联动开关量输出 |
| stChannelActParamInfo | 联动NVR预览、联动存储、联动抓拍 |
| stPresetActParamInfo | 联动云台预置位 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表)

### 通道联动

|  |
| --- |
| typedef struct tagNETDEVChannelActParamInfo  {  UINT32 udwNum;  INT32 adwChannelID[NETDEV\_CHANNEL\_MAX];  }NETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S,\*LPNETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 通道个数 |
| adwChannelID | 通道ID列表 |

**See also：**

[NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举)

### 使能联动参数

|  |
| --- |
| typedef struct tagNETDEVEnabledActParamInfo  {  BOOL bEnabled;  BYTE byRes[64];  }NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S, \*LPNETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 使能标记 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举)

### 联动云台预置位

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 联动动作数量 |
| stChannelPreset | 联动到预置位信息列表 |

**See also：**

[NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举)

### 联动云台预置位信息

|  |
| --- |
| typedef struct tagNETDEVChannelPreset  {  INT32 dwChannelID;  INT32 dwPresetID;  BYTE byRes[128];  }NETDEV\_CHANNEL\_PRESET\_S,\*LPNETDEV\_CHANNEL\_PRESET\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道号 |
| dwPresetID | 预置位编号 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_PRESET\_ACT\_PARAM\_INFO\_S](#_联动云台预置位)

### 联动开关量输出

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 联动的开关量输出个数 |
| astOutputAlarmStatusInfo | 联动的开关量输出列表 |

**See also：**

[NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举)

### 输出开关量的逻辑报警状态(手动告警)

|  |
| --- |
| typedef struct tagNETDEVOutputSwitchesAlarmStatus  {  INT32 dwBooleanId;  INT32 dwChannelId;  INT32 enAlarmStatus;  } NETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S, \*LPNETDEV\_OUTPUT\_SWITCH\_ALARM\_STATUS\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwBooleanId | 开关量编号 |
| dwChannelId | 通道ID,设备本身为0 |
| enAlarmStatus | 输出开关量报警状态参见#[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)

### 使能联动参数

|  |
| --- |
| typedef struct tagNETDEVEnabledActParamInfo  {  BOOL bEnabled;  BYTE byRes[64];  }NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S, \*LPNETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 使能标记 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_ALARM\_ACT\_ID\_E](#_使能联动参数枚举)

### 遮挡检测告警的配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 使能, 0:不使能1:使能 |
| udwSensitivity | 灵敏度，范围[1,100] |
| udwDuration | 持续时间，单位秒。长度范围[0, 10]，IPC需要 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 温度检测黑体信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| fTemperature | 黑体温度：取值范围：-40.00-150.00，精确到小数点后位小数，单位：摄氏度 |
| stAreaInfo | 矩形区域信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 温度检测矫正信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwMode | 校正模式0：体内，1：体表 |
| fCorrectionTemperature | 修正温度：取值范围：-10.00-10.00，精确到小数点后位小数，单位：摄氏度 |
| fEnvironmentTemperature | 环境温度：取值范围：-40.00-100.00，精确到小数点后位小数，单位：摄氏度 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 温度检测单位信息结构体

|  |
| --- |
| typedef struct tagNETDEVTempDetectioUnitInfo  {  INT32 dwUnit;  BYTE byRes[512];  }NETDEV\_TEMP\_DETECTION\_UNIT\_INFO\_S,\*LPNETDEV\_TEMP\_DETECTION\_UNIT\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwUnit | 温度单位0：摄氏度℃，1：华氏度℉，2：开尔文K |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 运动检测区域类型信息

|  |
| --- |
| typedef struct tagNETDEVMotionDetectionAreaType  {  UINT32 udwAreaType;  BYTE byRes[128];  }NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_S, \*LPNETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwAreaType | 区域类型参见枚举#[NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_E](#_运动检测区域类型枚举) |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 运动检测宏块区域信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 区域使能标识 |
| udwSensitivity | 运动检测灵敏度，值越大越灵敏，范围[1,100] |
| awGridInfo | 检测区域屏幕宏块信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 运动检测矩形区域信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 矩形区域个数 |
| astRectangleAreaInfoList | 矩形区域信息列表 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 运动检测矩形区域信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 矩形区域编号 |
| bEnabled | 区域使能标识 |
| udwSensitivity | 运动检测灵敏度，值越大越灵敏，范围[1,100] |
| udwDuration | 持续时间，范围[1,100] |
| udwTargetSize | 物体大小，范围[1,100] |
| stRectAreaInfo | 矩形区域信息,坐标值范围[0,9999] |
| byRes | 保留字段 |

**See also：**

[NETDEV\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_LIST\_S](#_运动检测矩形区域信息列表)

### 矩形区域坐标信息

|  |
| --- |
| typedef struct tagNETDEVRectAreaInfo  {  UINT32 udwTopLeftX;  UINT32 udwTopLeftY;  UINT32 udwBottomRightX;  UINT32 udwBottomRightY;  }NETDEV\_RECT\_AREA\_INFO\_S, \*LPNETDEV\_RECT\_AREA\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTopLeftX | x轴左点值 |
| udwTopLeftY | y轴顶点值 |
| udwBottomRightX | x轴右点值 |
| udwBottomRightY | y轴底点值 |

**See also：**

[NETDEV\_MOTION\_DETECTION\_AREA\_RECTANGLE\_INFO\_S](#_运动检测矩形区域信息)

### 开关量布控任务联动动作

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSwitchIndex | 开关量索引 |
| stLinkageActionList | 布控任务联动动作列表 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 阵列状态信息

|  |
| --- |
| typedef struct tagNETDEVRAIDStatus  {  BOOL bEnabled;  BYTE byRes[128];  }NETDEV\_RAID\_STATUS\_S, \*LPNETDEV\_RAID\_STATUS\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 阵列状态使能0:不使能1:使能 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 磁盘信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 硬盘个数 |
| astHDDInfo | 硬盘信息 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 硬盘信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 磁盘编号 |
| udwType | 磁盘类型参见枚举#[NETDEV\_HDD\_TYPE\_E](#_磁盘类型) |
| udwWorkMode | 磁盘工作模式参见枚举#[NETDEV\_HDD\_WORK\_MODE\_E](#_磁盘工作模式) |
| udwTotalCapacity | 硬盘总容量(MB) |
| szRAIDName | 阵列名称 |
| udwStatus | 磁盘状态参见枚举#[NETDEV\_HDD\_STATUS\_E](#_磁盘状态) |
| szManufacturer | 厂商名称 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_HDD\_INFO\_LIST\_S](#_磁盘信息列表)

### 存储容器信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwLocalHDDNum | 本地硬盘数量 |
| astLocalHDDList | 本地存储盘信息列表 |
| udwSDNum | SD卡数量 |
| astSDList | SD卡信息列表 |
| udwArrayNum | 阵列数量 |
| astArrayList | 阵列信息列表 |
| udwExtendCabinet1HDDNum | 拓展柜-1存储盘数量 |
| astExtendCabinet1HDDList | 拓展柜-1 信息列表 |
| udwExtendCabinet2HDDNum | 拓展柜-2存储盘数量 |
| astExtendCabinet2HDDList | 拓展柜-2 信息列表 |
| udwNASNum | NAS数量 |
| astNASList | NAS信息列表 |
| udweSATANum | eSATA硬盘数量 |
| asteSATAList | eSATA信息列表 |
| udwIPSANNum | IPSAN数量 |
| astIPSANList | IPSAN信息列表 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 存储容器信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 磁盘编号 |
| udwRemainCapacity | 存储容器剩余容量(MB) |
| udwTotalCapacity | 存储容器总容量(MB) |
| szManufacturer | 厂商名称 |
| udwStatus | 存储容器状态参见枚举#[NETDEV\_STORAGE\_CONTAINER\_STATUS\_E](#_存储容器状态) |
| udwProperty | 存储盘属性,当udwStatus为时无效参见枚举  #[NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_E](#_存储容器属性) |
| udwFormatProgress | 格式化进度，百分比 |
| udwGroupID | 盘组序号 |
| udwTemperature | 硬盘温度(摄氏度) |
| byRes | 保留字节 |

**See also：**

[NETDEV\_STORAGE\_CONTAINER\_INFO\_LIST\_S](#_存储容器信息列表)

### 扩展存储容器信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 磁盘编号 |
| udwRemainCapacity | 存储容器剩余容量(MB) |
| udwTotalCapacity | 存储容器总容量(MB) |
| udwAddressType | IP地址类型,参见枚举#[NETDEV\_ADDR\_TYPE\_E](#_地址类型) |
| szIPAddress | 服务器IP地址 |
| szPath | NAS服务器存储路径 |
| udwUsageType | 用途,参见枚举#[NETDEV\_STORAGE\_CONTAINER\_USAGE\_TYPE\_E](#_存储容器用途) |
| udwStatus | 存储容器状态参见枚举#[NETDEV\_STORAGE\_CONTAINER\_STATUS\_E](#_存储容器状态) |
| udwProperty | 存储盘属性,当udwStatus为时无效参见枚举  #[NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_E](#_存储容器属性) |
| udwFormatProgress | 格式化进度，百分比 |
| udwGroupID | 盘组序号 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_STORAGE\_CONTAINER\_INFO\_LIST\_S](#_存储容器信息列表)

### 硬盘Smart信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | IN 存储容器编号 |
| szManufacturer | 厂商名称 |
| udwTemperature | 温度(℃) |
| szDeviceModel | 硬盘型号 |
| udwUsedDays | 使用天数 |
| udwHealthAssessment | 整体评估结果参见枚举  #[NETDEV\_HDD\_HEALTH\_ASSESSMENT\_STATUS\_E](#_硬盘健康评估状态) |
| szFirmware | 硬盘固件版本 |
| udwSmartNum | Smart详情项数量 |
| SmartDetailsInfoList | Smart详情项列表 |
| bCheckResult | 自我评估结果, TRUE 通过，FALSE 未通过 |
| udwCheckPrograss | 检测进度[0,100] |
| udwCheckStatus | 检测状态参见枚举#[NETDEV\_HDD\_SMART\_CHECK\_STATUS\_E](#_硬盘Smart检测状态) |
| udwCheckType | 检测类型参见枚举#[NETDEV\_HDD\_SMART\_CHECK\_TYPE\_E](#_硬盘Smart检测类型) |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 硬盘Smart检测信息

|  |
| --- |
| typedef struct tagNETDEVHDDSmartCheckInfo  {  UINT32 udwID;  UINT32 udwType;  BYTE byRes[128];  }NETDEV\_HDD\_SMART\_CHECK\_INFO\_S,\*LPNETDEV\_HDD\_SMART\_CHECK\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 存储容器编号 |
| udwType | 检测类型参见枚举# [NETDEV\_HDD\_SMART\_CHECK\_TYPE\_E](#_硬盘Smart检测类型) |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 照片接收服务器配置信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 照片服务器数量当前支持数量为2 |
| astPhotoServerList | 照片服务器配置列表 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 功能增强配置信息

|  |
| --- |
| typedef struct tagNETDEVEnhancedConfigInfo  {  BOOL bEnabledStorage;  BYTE byRes[512];  }NETDEV\_ENHANCED\_CONFIG\_INFO\_S, \*LPNETDEV\_ENHANCED\_CONFIG\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabledStorage | UNP/云设备存储开关使能状态，0:关闭，1:开启 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 智能服务器配置列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 管理服务器个数当前支持数量为2 |
| astDataServerList | 管理服务器列表 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 智能服务器配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwIndex | 管理服务器索引从0开始 |
| udwMngProtocol | 通信协议类型见# [NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_E](#_管理服务器接入协议类型) |
| szDeviceID | 前端设备的设备ID[1, 32] |
| szServerID | 注册服务器ID [1, 32] |
| szDeviceName | 设备名称[0, 20] |
| szRegPassword | 注册服务器密码(0, 16] |
| szVideoChlID | 视频通道ID[0,32] |
| szAudioChlID | 音频通道ID[0,32] |
| udwExpire | 注册有效期[3600,36000]秒 |
| szAddress | 服务器IPv4地址 |
| udwPort | 服务器端口 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_DATA\_SERVER\_LIST](#_智能服务器配置列表)

### 服务器用户信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szUserName | 用户名，建议范围[1,32] |
| bModifyPasswd | 是否需修改密码0:不修改1:修改 |
| szPasswd | 平台接入密码，建议范围[0,32] 空密码则下发空字符串 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_MANAGER\_SERVER\_INFO\_S](#_智能服务器配置信息_1)

### 运动检测告警参数

|  |
| --- |
| typedef struct tagNETDEVMotionIntervalInfo  {  UINT32 udwSuppressTime;  UINT32 udwClearTime;  BYTE byRes[512];  }NETDEV\_MOTION\_INTERVAL\_INFO\_S, \*LPNETDEV\_MOTION\_INTERVAL\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSuppressTime | 报警抑制间隔时间,单位:秒,范围[5, 3600] |
| udwClearTime | 报警恢复间隔时间,单位:秒,范围[1,600] |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 管理服务器配置列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 管理服务器个数当前支持数量为2 |
| astManagerServerList | 管理服务器列表 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 可视域参数信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dAzimuthHorizontal | 水平方位角度（可视域在水平界面上投影的夹角的中轴线与水平面基准轴的角的角度）单位:弧度表示范围:[-1.0000,+1.0000] |
| dAzimuthVertical | 垂直方位角度（可视域在垂直面上投影的夹角的中轴线与垂直面上基准轴的角的角度）单位:弧度表示精确到小数点后位范围:[-1.0000,+1.0000] |
| dInclinationHorizontal | 水平倾斜角度（设备水平基准面与水平面的夹角）单位:弧度表示精确到小数点后位范围:[-1.0000,+1.0000] |
| dAngelHorizontal | 水平可视角度单位:度，精确到小数点后4位 |
| dAngelVertical | 垂直可视角度单位:度，精确到小数点后4位 |
| dDistance | 可视距离单位:米，精确到小数点后4位 |
| dInstallationHeight | 安装高度单位：米，精确到小数点后4位 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 验证信息

|  |
| --- |
| typedef struct tagNETDEVResetPwdVertifyInfo  {  CHAR szEMail[NETDEV\_LEN\_256]; /\* 邮箱\*/  CHAR szPhone[NETDEV\_LEN\_256]; /\* 电话号码\*/  BYTE byRes[256]; /\* 保留字段\*/  }NETDEV\_RESET\_PWD\_VERTIFY\_INFO\_S, \*LPNETDEV\_RESET\_PWD\_VERTIFY\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szEMail | 邮箱 |
| szPhone | 电话号码 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 存储配额信息

|  |
| --- |
| typedef struct tagNETDEVStorageQuotaInfo存储配额信息  {  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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwStorageCapUnit | 存储容量单位(默认为GB),参见#[NETDEV\_STORAGE\_CAP\_UNIT\_E](#_存储容量单位) |
| dwRecUsedCap | 通道录像已用容量 |
| dwPicUsedCap | 通道抓图已用容量 |
| dwAIPicUsedCap | 通道智能抓拍照片已用容量，IPC必选 |
| dwUsedCap | 通道存储空间已使用容量，IPC必选 |
| dwTotalCap | 通道存储空间总容量 |
| dwRecQuota | 通道录像配额容量 |
| dwPicQuota | 通道抓图配额容量 |
| dwAIPicQuota | 通道智能抓拍图片配额容量，IPC必选 |
| dwGroupID | 盘组ID号 |
| dwGroupTotalCap | 所选牌组的总容量 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 自定义协议信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTransport | 传输模式, 参见枚举[NETDEV\_TRANS\_PROTOCOL\_E](#_传输协议) |
| udwPort | 端口 |
| stCustomProSimpleInfo | 自定义协议配置信息 |
| stMainStream | 主流协议信息 |
| stSubStream | 辅流协议信息 |
| stThirdStream | 三流协议信息 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 自定义协议配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwProID | 自定义协议号, 必填 |
| szProName | 自定义协名称[0, 64] |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CUSTON\_PROTOCOL\_INFO\_S](#_自定义协议信息)

### 自定义流协议信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwEnable | 1:使能; 0:不使能 |
| szStreamURL | 实况URL地址[0, 128] |
| szPlaybackStreamURL | 回放URL地址[0, 128] |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CUSTON\_PROTOCOL\_INFO\_S](#_自定义协议信息)

### 自定义协议配置单例信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 自定义协议数量, 当前最大支持64 |
| astCustomProSimpleList | 自定义协议列表 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 自定义协议配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwProID | 自定义协议号, 必填 |
| szProName | 自定义协名称[0, 64] |
| byRes | 保留字节 |

**See also：**

[NETDEV\_CUSTON\_PROTOCOL\_SIMPLE\_INFO\_LIST\_S](#_自定义协议配置单例信息列表)

### 存储策略信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwStorageRes | 存储资源, IPC必选,参见#[NETDEV\_STORAGE\_RES\_TYPE\_E](#_存储资源) |
| stRecStoreStrategyInfo | 视频存储策略信息 |
| stPicStoreStrategyInfo | 普通抓拍图片存储策略信息 |
| stAIPicStoreStrategyInfo | 智能抓拍图片存储策略信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 视频存储策略信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bStoregeEnabled | 存储使能,0:开启,1:关闭 |
| dwFullStrategy | 慢策略参数，默认满覆盖，参见枚举#[NETDEV\_FULL\_STRATEGY\_TYPE\_E](#_满存储策略) |
| dwPlanStreamType | 计划存储流类型，参见#[NETDEV\_LIVE\_STREAM\_INDEX\_E](#_实况业务流索引) |
| dwManualStreamType | 手动存储流类型，参见#[NETDEV\_LIVE\_STREAM\_INDEX\_E](#_实况业务流索引) |
| dwEventStreamType | 告警存储流类型，参见#[NETDEV\_LIVE\_STREAM\_INDEX\_E](#_实况业务流索引) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_STORAGE\_STRATEGY\_S](#_存储策略信息)

### 图片存储策略信息

|  |
| --- |
| typedef struct tagNETDEVPicStorageStrategyInfo  {  INT32 dwStorageMode;  INT32 dwFullStrategy;  BYTE byRes[256];  }NETDEV\_PIC\_STORAGE\_STRATEGY\_INFO\_S, \*LPNETDEV\_PIC\_STORAGE\_STRATEGY\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwStorageMode | 存储模式，参见枚举#[NETDEV\_PIC\_STORAGE\_MODE\_E](#_图片存储模式) |
| dwFullStrategy | 满策略参数，默认满覆盖，参见枚举#[NETDEV\_FULL\_STRATEGY\_TYPE\_E](#_满存储策略) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_STORAGE\_STRATEGY\_S](#_存储策略信息)

### 后焦调节参数信息

|  |
| --- |
| typedef struct tagNETDEVBackFocusInfo  {  UINT32 udwCmd;  UINT32 udwPara1;  BYTE byRes[256];  }NETDEV\_BACKFOCUS\_INFO\_S, \*LPNETDEV\_BACKFOCUS\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwCmd | 操作命令字见# [NETDEV\_BACKFOCUS\_CMD\_E](#_后焦控制操作命令) |
| udwPara1 | 操作参数见# [NETDEV\_BACKFOCUS\_CMD\_E](#_后焦控制操作命令) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 智能属性配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stFaceCfg | 人脸属性配置 |
| stTempCfg | 温度属性配置 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 人脸属性配置

|  |
| --- |
| typedef struct tagNETDEVFaceConfig  {  UINT32 udwCondition;  BYTE byRes[512];  }NETDEV\_FACE\_CFG\_S, LPNETDEV\_FACE\_CFG\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwCondition | 报警条件0：未戴(包括未知) ，1：戴 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SMART\_ATTR\_CFG\_S](#_智能属性配置)

### 温度属性配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwCondition | 报警条件0：低于，1：高于，2：匹配 |
| szThreshold | 阈值温度单位：摄氏度，精度：小数点后2位 |
| szdwRange | 温度允许误差单位：摄氏度 精度：小数点后2位 范围：[0.00, 10.00] |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SMART\_ATTR\_CFG\_S](#_智能属性配置)

### 图片矫正参数

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 是否开启0：关闭1：开启 |
| udwFaceWidth | 生成的人脸宽度范围：[120,150] 单位：px |
| udwPupilDistance | 瞳距范围：[120,150] 单位：px |
| udwMinImageSize | 图片最小值单位：KB |
| udwMaxImageSize | 图片最大值单位：KB |
| fBlackAndWhiteThreshold | 黑白阈值范围：-1 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 设备时间同步方式

|  |
| --- |
| typedef struct tagNETDEVTimeSyncMode  {  UINT32 udwMode;  BYTE byRes[256];  }NETDEV\_TIME\_SYNCMODE\_S, \*LPNETDEV\_TIME\_SYNCMODE\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwMode | 时间同步方式，详见枚举#[NETDEV\_TIME\_SYNCMODE\_E](#_时间同步方式枚举) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 场景自动切换开关信息

|  |
| --- |
| typedef struct tagNETDEVSceneAutoSwitchInfo  {  BOOL bAutoSwitchEnable;  BYTE byRes[256];  }NETDEV\_SCENE\_AUTO\_SWITCH\_INFO\_S, \*LPNETDEV\_SCENE\_AUTO\_SWITCH\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bAutoSwitchEnable | 是否开启自动切换0：关闭1：开启 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 场景模板信息

|  |
| --- |
| typedef struct tagNETDEVSceneTemplateInfo  {  UINT32 udwID;  BYTE byRes[256];  }NETDEV\_SCENE\_TEMPLATE\_INFO\_S, \*LPNETDEV\_SCENE\_TEMPLATE\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 场景模板编号，从0开始 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 所有场景信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 场景个数 |
| astSceneInfo | 场景信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 场景信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szName | 场景名称 |
| bAutoSwitchEnable | 是否开启自动切换0：关闭1：开启 |
| udwID | 场景模板编号 |
| udwType | 场景类型 |
| udwPriority | 优先级，范围[1, SceneTemplateNum]，最大值是可支持的场景模板数量 |
| udwTriggerNum | 场景自动切换触发条件个数 |
| astTriggerInfo | 场景自动切换触发条件详细信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SCENE\_INFO\_LIST\_S](#_所有场景信息)

### 场景自动切换触发条件详细信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szBegin | 开始时间 |
| szEnd | 结束时间 |
| udwEnvironmentNum | 环境参数个数 |
| astEnvironmentInfo | 环境参数信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SCENE\_INFO\_S](#_场景信息)

### 环境参数信息

|  |
| --- |
| typedef struct tagNETDEVEnvParamInfo  {  UINT32 udwType;  UINT32 udwMin;  UINT32 udwMax;  BYTE byRes[256];  }NETDEV\_ENV\_PARAM\_INFO\_S, \*LPNETDEV\_ENV\_PARAM\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 环境参数类型 |
| udwMin | 最小值，不能大于最大值 |
| udwMax | 最大值，不能小于最小值 |
| byRes | 保留字段 Reserved |

**See also：**

[NETDEV\_TRIGGER\_DETAIL\_INFO\_S](#_场景自动切换触发条件详细信息)

### 属性采集信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bAttributeAll | 是否全属性0：非全属性1：全属性 |
| stFaceAttrInfo | 人脸属性开关信息 |
| stPedAttrInfo | 行人属性开关信息 |
| stNoMotorVehAttrIofo | 非机动车属性开关信息 |
| stMotorVehAttrInfo | 车辆属性开关信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 人脸属性开关信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bIsEnableGender | 性别检测开关0：关闭1：开启 |
| bIsEnableAge | 年龄检测开关0：关闭1：开启 |
| bIsEnableGlasses | 眼镜检测开关0：关闭1：开启 |
| bIsEnableMaskFlag | 是否戴口罩检测开关0：关闭1：开启 |
| bIsEnableTemp | 温度检测开关0：关闭1：开启 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_ATTR\_COLLECT\_INFO\_S](#_属性采集信息)

### 行人属性开关信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bIsEnableGender | 性别检测开关0：关闭1：开启 |
| bIsEnableAge | 年龄检测开关0：关闭1：开启 |
| bIsEnableMaskFlag | 是否戴口罩检测开关0：关闭1：开启 |
| bIsEnableUpperColor | 上衣颜色检测开关0：关闭1：开启 |
| bIsEnableLowerColor | 下衣颜色检测开关0：关闭1：开启 |
| bIsEnableUpClothesTexture | 上衣纹理检测开关0：关闭1：开启 |
| bIsEnableUpperClothingStyle | 上衣款式检测开关0：关闭1：开启 |
| bIsEnableLowerClothingStyle | 下衣款式检测开关0：关闭1：开启 |
| bIsEnableShoes | 鞋子检测开关0：关闭1：开启 |
| bIsEnableBodyAngle | 身姿角度检测开关0：关闭1：开启 |
| bIsEnableBagInfo | 携包信息检测开关0：关闭1：开启 |
| bIsEnableHairStyle | 发型检测开关0：关闭1：开启 |
| bIsEnableDirection | 运动方向检测开关0：关闭1：开启 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_ATTR\_COLLECT\_INFO\_S](#_属性采集信息)

### 非机动车属性开关信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bIsEnableGender | 性别检测开关0：关闭1：开启 |
| bIsEnableAge | 年龄检测开关0：关闭1：开启 |
| bIsEnableUpperColor | 上衣颜色检测开关0：关闭1：开启 |
| bIsEnableUpperClothingStyle | 上衣款式检测开关0：关闭1：开启 |
| bIsEnableNonVehicleType | 非机动车车型检测开关0：关闭1：开启 |
| bIsEnableDirection | 行驶方向检测开关0：关闭1：开启 |
| bIsEnableSpeed | 移动速度检测开关0：关闭1：开启 |
| bEnablePlateNumber | 车牌号码检测开关0：关闭1：开启 |
| bEnablePlateType | 车牌类型检测开关0：关闭1：开启 |
| bEnablePlateColor | 车牌颜色检测开关0：关闭1：开启 |
| bEnableLaneNo | 车道号检测开关0：关闭1：开启 |
| bEnablePeccancy | 违法类型检测开关0：关闭1：开启 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_ATTR\_COLLECT\_INFO\_S](#_属性采集信息)

### 车辆属性开关信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bIsEnableLogo | 车辆品牌检测开关0：关闭1：开启 |
| bIsEnableVehicleType | 车辆类型检测开关0：关闭1：开启 |
| bIsEnableVehicleColor | 车辆颜色检测开关0：关闭1：开启 |
| bIsEnablePlateNumber | 车牌号码检测开关0：关闭1：开启 |
| bIsEnablePlateType | 车牌类型检测开关0：关闭1：开启 |
| bIsEnablePlateColor | 车牌颜色检测开关0：关闭1：开启 |
| bIsEnableDirection | 行驶方向检测开关0：关闭1：开启 |
| bIsEnableSpeed | 行驶速度检测开关0：关闭1：开启 |
| bEnableSunVisor | 遮阳板状态检测开关0：关闭1：开启 |
| bEnableSeatBelt | 安全带状态检测开关0：关闭1：开启 |
| bEnableDriverMobile | 主驾驶开车打电话状态检测开关0：关闭1：开启 |
| bEnableAim | 年检贴状态检测开关0：关闭1：开启 |
| bEnablePendant | 挂坠状态检测开关0：关闭1：开启 |
| bEnableLaneNo | 车道号检测开关0：关闭1：开启 |
| bEnablePeccancy | 违法类型检测开关0：关闭1：开启 |
| bEnableYellowPlateMark | 黄标车标志状态检测开关0：关闭1：开启 |
| bEnableVehicleBrandType | 车款检测开关0：关闭1：开启 |
| bEnableDangerousGoodsMark | 危险品标志检测开关0：关闭1：开启 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_ATTR\_COLLECT\_INFO\_S](#_属性采集信息)

### 时间配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwTimeZone | 时区参见枚举# [NETDEV\_TIME\_ZONE\_E](#_时区) |
| stTime | 时间 |
| bEnableDST | 夏令时使能 |
| stTimeDSTCfg | 夏令时配置 |
| udwDateFormat | 日期格式0：YYYY-MM-DD 年月日1：MM-DD-YYYY 月日年2：DD-MM-YYYY 日月年 |
| udwHourFormat | 时间格式0 ：小时制 1:24 小时制 |
| byRes | 保留字段 |

### 时间参数

|  |
| --- |
| typedef struct tagNETDEVTime  {  INT32 dwYear;  INT32 dwMonth;  INT32 dwDay;  INT32 dwHour;  INT32 dwMinute;  INT32 dwSecond;  }NETDEV\_TIME\_S, \*LPNETDEV\_TIME\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwYear | 年 |
| dwMonth | 月 |
| dwDay | 日 |
| dwHour | 时 |
| dwMinute | 分 |
| dwSecond | 秒 |

**See also：**

[NETDEV\_TIME\_CFG\_S](#_时间配置_3)

### 系统IP地址信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | NTP服务器使能0：不开启 1：开启 |
| ulAddressType | 地址类型0：IPv4 1：IPv6(暂不支持) 2: 域名（NVR和一体机支持） |
| szIPAddress | NTP服务器IP地址，字符长度范围[0, 64] , ulAddressType为0时，该节点必选 |
| szDomainName | NTP服务器域名，字符长度范围[ [0,64], ulAddressType为2时，该节点必选 |
| ulPort | NTP端口，范围[1-65535] IPC不支持配置 |
| ulSynchronizeInterval | 更新间隔NVR、VMS支持范围：5/10/15/30分钟，1/2/3/6/12h，1天，1周。IPC支持范围：30-3600s，以上时间周期，都需要转换为秒为单位的时间值 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SYSTEM\_NTP\_INFO\_LIST\_S](#_NTP列表)

### 相对于电视墙的位置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stTopLeft | 左上角的坐标点 |
| stBottomRight | 右下角的坐标点 |

**See also：**

[NETDEV\_TEMP\_DETECTION\_BLACKBODY\_INFO\_S](#_温度检测黑体信息结构体)

### 坐标信息

|  |
| --- |
| typedef struct tagNETDEVXWArea  {  UINT32 udwX;  UINT32 udwY;  }NETDEV\_XW\_COORD\_S, \*LPNETDEV\_XW\_COORD\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwX | X轴坐标 |
| udwY | Y轴坐标 |

**See also：**

[NETDEV\_XW\_AREA\_S](#_相对于电视墙的位置信息)

### 硬盘Smart详细信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwAttributeID | 属性ID |
| szAttributeName | 属性名称 |
| udwStatus | 状态参见枚举#[NETDEV\_HDD\_SMART\_ASSESSMENT\_STATUS\_E](#_硬盘Smart评估状态) |
| udwHex | 显示为十六进制 |
| udwThresh | 阈值 |
| udwCurrentValue | 当前值 |
| udwWorstValue | 最差值 |
| udwActualValue | 实际值 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_HDD\_SMART\_INFO\_S](#_硬盘Smart信息)

### 照片接收服务器配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwProtocol | 通信协议类型参见：[NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_E](#_照片服务器通信协议类型) |
| udwSubProtocol | 通信协议子类型参见：[NETDEV\_PHOTO\_SERVER\_SUBPRO\_TYPE\_E](#_照片服务器通信子协议类型) |
| udwPort | TMS服务器端口号[0,65535] |
| szIPV4Address | TMS服务器IPv4地址 |
| szTollgateID | 卡口编号(0, 36]; 可选字段，接口不携带该字段时表示不更新该字段 |
| szCameraID | 相机编号(0, 36];可选字段，接口不携带该字段时表示不更新该字段 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_PHOTO\_SERVER\_CFG\_INFO\_LIST\_S](#_照片接收服务器配置信息列表)

### 智能服务器配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwIndex | 智能服务器索引从开始 |
| bEnable | 智能服务器使能开关 |
| udwProtocol | 通信协议类型目前只支持：1:TCP传输，见# [NETDEV\_DATA\_SERVER\_PROTOCOL\_TYPE\_E](#_智能服务器接入协议类型) |
| szCameraID | 相机编号(0, 36] 长度为代表不修改 |
| szTollgateID | 卡口编号(0, 36] 长度为代表不修改 |
| szServerAddr | 服务器IPv4地址 |
| udwServerPort | 服务器端口 |
| szDrivewayID | 道路编号IPC支持 |
| szDirectionID | 方向编号IPC支持 |
| szDeviceID | 设备编号IPC支持 |
| szAreaId | 管辖区域IPC支持 |
| stServerUserInfo | 用户信息 |
| stViidCfg | 视图库信息，参见 [NETDEV\_VIID\_CFG\_S](#_视图库信息结构体) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_DATA\_SERVER\_LIST](#_智能服务器配置列表)

### 通道OSD所有内容

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | OSD区域数量 |
| astContentList | OSD区域内容信息列表 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 通道OSD内容信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | OSD区域使能 |
| udwOSDID | OSD区域序号，范围[0,7] |
| udwAreaOSDNum | 当前区域内OSD数目 |
| udwTopLeftX | OSD区域横坐标,范围[0,9999] |
| udwTopLeftY | OSD区域纵坐标,范围[0,9999] |
| astContentInfo | 当前区域内OSD内容信息 |

**See also：**

[NETDEV\_OSD\_CONTENT\_S](#_通道OSD所有内容)

### 内容信息

|  |
| --- |
| typedef struct tagNETDEVContentInfo  {  UINT32 udwContentType;  CHAR szOSDText[NETDEV\_OSD\_TEXT\_MAX\_LEN];  }NETDEV\_CONTENT\_INFO\_S,\*LPNETDEV\_CONTENT\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwContentType | OSD内容类型,参见枚举[NETDEV\_OSD\_CONTENT\_TYPE\_E](#_OSD内容类型) |
| szOSDText | OSD文本信息 |

**See also：**

[NETDEV\_OSD\_CONTENT\_INFO\_S](#_通道OSD内容信息)

### 温度检测告警的配置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 规则类型：0: 全屏规则1: 普通规则2: 普通、对比规则3: 人脸测温规则 |
| stEntireImageRuleInfo | 全屏规则信息，当Type值为0时，此项必选 |
| stCommonRuleInfo | 普通规则信息，当Type值为1或2时，此项必选 |
| stCompareRuleInfo | 对比规则信息，当Type值为2时，此项必选 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CONFIG\_COMMAND\_E](#_设备配置命令)

### 温度检测基本信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 告警类型0：最高温，1：最低温，：2平均温，3：温度差值，4：温度变化率（注：1、当为对比规则时，告警类型只支持0,1和2；2、当区域类型为点时，告警类型只支持2和4） |
| dwCondition | 报警条件0：低于，1：高于，2：匹配 |
| fThreshold | 阈值温度，范围[-20,150]，默认0，单位：度（告警类型不为温度变化率时此项必选），精确到小数点后2位 |
| fChangeRate | 温度变化率，范围[-170,170]，单位：度/分钟（告警类型为温度变化率时此项必选），精确到小数点后两位 |
| fRange | 温度允许误差，范围[0,10]，单位：度，精确到小数点后两位 |
| udwDuration | 持续时间，范围[1,100]，单位：秒 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_TEMP\_DETECTION\_RULE\_INFO\_S](#_温度检测告警的配置信息)

### 温度检测通用信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 温度告警普通规则数量，从0开始，范围[0,12] |
| astTempCommRuleInfo | 温度检测通用信息结构体 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_TEMP\_DETECTION\_RULE\_INFO\_S](#_温度检测告警的配置信息)

### 温度检测通用信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 规则编号，从开始，范围[0,5]。Get和PUT方法时必带 |
| bEnabled | 使能状态0：关闭，1：开启 |
| szName | 规则名称，范围[0,20] |
| udwAreaType | 规则区域类型0：线，4：多边形（最大支持六边形，与智能规则框保持一致），7：点 |
| udwAreaPointNum | 检测区域图形的顶点个数1：点，2：直线，3-6：多边形 |
| astAreaPoint | 检测区域图形各顶点坐标 |
| stTempBaseInfo | 温度检测基本信息结构体 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_TEMP\_DETECTION\_COMM\_RULE\_LIST\_S](#_温度检测通用信息列表)

### 温度检测比较信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 温度告警比较规则数量，从0开始，范围[0,12] |
| astTempCompareRuleInfo | 温度检测比较信息结构体 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_TEMP\_DETECTION\_RULE\_INFO\_S](#_温度检测告警的配置信息)

### 温度检测比较信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 规则编号，从0开始，范围[0,5]。Get和PUT方法时必带 |
| bEnabled | 使能状态0：关闭,1：开启 |
| szName | 规则名称，范围[0,20] |
| udwFirstCompareRuleID | 第一个对比规则的序号，范围[1,12] |
| udwSecondCompareRuleID | 第二个对比规则的序号，范围[1,12]（与第一个规则号不可以相同） |
| stTempBaseInfo | 温度检测基本信息结构体 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_TEMP\_DETECTION\_COMP\_RULE\_LIST\_S](#_温度检测比较信息列表)

### 设备基本信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stDevAddr | 设备IP地址信息 |
| stDevUserInfo | 设备用户名、密码 |
| szDevName | 设备名称 |
| szDevDesc | 设备描述 |
| szDevModel | 设备型号 |
| dwDevID | 设备ID |
| dwDevStatus | 设备状态, 参考# [NETDEV\_DEVICE\_STATUS\_E](#_设备状态) |
| dwDevType | 设备类型，参考# [NETDEV\_DEVICE\_MAIN\_TYPE\_E](#_设备类型枚举定义) |
| dwDevSubType | 设备子类型，参考# [NETDEV\_DEVICE\_SUB\_TYPE\_E](#_设备子类型) |
| dwOrgID | 组织编号 |
| dwAccessProtocol | 接入协议参考# [NETDEV\_ACCESS\_PROTOCOL\_E](#_接入协议) |
| dwAccessMode | 接入方式参考# [NETDEV\_DEVICE\_ACCESS\_MODE\_E](#_接入方式) |
| dwServerID | 所属服务器ID |
| dwAudioResID | 音频通道ID |
| dwIsPTZNeeded | 是否需要云台0: 不需要1: 需要255: 自适应 |
| szVIIDCode | 视图库编码,仅卡口设备时有效 |
| dwVIIDStatus | 视图库状态，用来标识当前设备是否已通过视图库协议连接，0：离线1：在线 |
| szSerialNum | 设备序列号 |
| szSoftVersion | 软件版本号 |
| szMacAddr | MAC地址 |
| dwStoreStatus | 设备存储状态。0: 空闲1: 未格式化2: 格式化中3: 运行中 |
| stOnvifInfo | onvif信息 |
| stGBInfo | 国标信息当AccessProtocol值为3时该节点必选，其他可选 |
| pstSmartLockInfo | 锁设备信息需调用者分配内存 |
| szManufacture | 厂商名称 |
| szDeviceCode | 设备编码[1,32] 添加播放盒时必选 |
| pstPlayerInfo | 播放盒信息当Type为11时必选需要malloc分配内存 |
| udwCustomProtocolID | 自定义协议ID，当AccessProtocol值为4时该节点必选 |
| udwChlMaxNum | 设备通道号最大数量，当AccessProtocol值为4时该节点必选 |
| udwChlIndexNum | 设备通道号数量，当AccessProtocol值为4时该节点必选，上限255 |
| pudwChlIndexList | 通道号列表，需动态分配内存，建议分配个256个 |
| dwImageProtocol | 图片协议，设备类型Type为5智能设备时必带1:私有2:视图库 |

**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地址信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwType | 地址类型，参见枚举[NETDEV\_IP\_ADDRESS\_TYPE\_E](#_IP地址类型) |
| szIPAddr | IP地址/域名 |
| dwPort | 端口号 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1)

### 用户账号信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szUserName | 用户名 |
| szPassword | 密码 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1)

### Onvif信息

|  |
| --- |
| typedef struct tagstNETDEVOnvifInfo  {  UINT32 udwTransportMode;  BYTE byRes[128];  }NETDEV\_ONVIF\_INFO\_S, \*LPNETDEV\_ONVIF\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTransportMode | 传输模式，参见枚举[NETDEV\_TRANS\_PROTOCOL\_E](#_传输协议) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1)

### 国标信息

|  |
| --- |
| typedef struct tagNETDEVGBInfo  {  CHAR szUniCode[NETDEV\_LEN\_32];  UINT32 udwTransport;  BYTE byRes[128];  }NETDEV\_GBINFO\_S, \*LPNETDEV\_GBINFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szUniCode | 国标资源编码，范围[1, 32] |
| udwTransport | 传输模式0: TCP 1: UDP |
| byRes | 保留字节 |

**See also：**

[NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1)

### 智能锁信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 锁类型0: WIFI锁1: NBIoT锁 |
| udwSignal | 锁信号详见[NETDEV\_LOCK\_SIGNAL\_E](#_锁信号) |
| udwStatus | 锁状态0：在线 1：离线 |
| udwBatteryPercent | 电量，取值范围[0,100] |
| szSN | 锁设备序列号，字符串范围[0,20] |
| szIMEI | 国际移动设备识别码Type为时携带,字符串长度范围[1,16] |
| szVersion | 锁版本信息字符串长度范围[1,64] |
| szRoomName | 绑定房间名称字符串长度范围[1, 128] |
| byRes | 保留字节 |

**See also：**

[NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1)

### 播放盒信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPlayerID | 播放盒ID |
| udwVendor | 厂商添加播放盒时必选 |
| udwModel | 型号添加播放盒时必选 |
| szPlayerName | 播放盒名称[1,64] |
| udwOrgID | 组织ID 播放盒信息查询时必选 |
| szOrgName | 组织名称[1,64] 播放盒信息查询时必选 |
| szPlayerMngtServerName | 播放盒管理设备名称[1, 64] 播放盒信息查询时必选 |
| szPlayerMngtServerIP | 播放盒管理设备IP 播放盒信息查询时必选 |
| stResolution | 屏幕像素 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_DEV\_BASIC\_INFO\_S](#_设备基本信息_1)

### 分辨率信息

|  |
| --- |
| typedef struct tagNETDEVXWVideoResolution  {  UINT32 udwWidth;  UINT32 udwHeight;  }NETDEV\_XW\_RESOLUTION\_S, \*LPNETDEV\_XW\_RESOLUTION\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwWidth | 宽 |
| udwHeight | 高 |

**See also：**

[NETDEV\_IPM\_PLAYER\_BASIC\_INFO\_S](#_播放盒信息_1)

### 设备通道信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道号 |
| szDevModel | 设备型号 |
| szSerialNum | 硬件序列号 |
| szFirmwareVersion | 软件版本号 |
| szHardewareID | 硬件标识 |
| szUbootVersion | UBOOT引导版本 |
| byRes | 保留字 |

**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)

### 设备XW本地信号源通道信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stChnBaseInfo | 通道基本信息 |
| dwVideoSrc | 视频输入格式[NETDEV\_VIDEO\_SRC\_TYPE\_E](#_视频输入类型枚举) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CHN\_TYPE\_E](#_通道类型)

### 设备通道基本信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道ID |
| dwDevID | 设备ID |
| dwOrgID | 组织ID |
| dwChnType | 通道类型，参见[NETDEV\_CHN\_TYPE\_E](#_通道类型) |
| dwChnStatus | 通道状态, 参见[NETDEV\_CHN\_STATUS\_E](#_编码通道状态) |
| dwChnIndex | 通道索引 |
| szChnName | 通道名称 |
| udwRight | 通道权限 |
| udwAbnormalReason | 视频通道异常的原因 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_DEV\_CHN\_XW\_ENCODE\_INFO\_S](#_设备XW本地信号源通道信息)

### 设备基础信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChlID | 视频输入通道号 |
| dwDeviceType | 设备类型，参考枚举[NETDEV\_DEVICE\_TYPE\_E](#_错误码列表) |
| dwDevCode | 设备编号 |
| dwCurrentLang | 设备当前语言参见枚举#[NETDEV\_LANG\_TYPE\_E](#_语言类型枚举) |
| stBuildDate | 发布时间 |
| szSerialNum | 设备序列号 |
| szMacAddress | IPv4的Mac地址 |
| szDevName | 设备名称 |
| szDevModel | 设备型号 |
| szVIIDVersion | 视图库组件版本号 |
| szSoftwareVersion | 内部软件版本 |
| szFirmwareVersion | 软件版本号 |
| szHardewareID | 硬件标识 |
| szUbootVersion | UBOOT引导版本 |
| szManufacturer | 厂商名称，字符串长度[1,64] |
| stClientVersionsInfo | 版本信息，可选：仅一体机支持 |
| szProtoName | 设备原型机名称 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_GetDeviceBaseInfo](#_获取设备基础信息)

### 时间信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwYear | 所在年份 |
| dwMonth | 所在月份 |
| dwDay | 具体日期 |
| dwHour | 具体小时 |
| dwMinute | 具体分钟 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_DEVICE\_BASE\_INFO\_S](#_设备基础信息)

### 客户端版本信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 数量 |
| pstClientVersionsInfo | 客户端版本信息列表 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_DEVICE\_BASE\_INFO\_S](#_设备基础信息)

### 客户端版本信息

|  |
| --- |
| typedef struct tagstNETDEVClientVersionsInfo  {  UINT32 udwType;  CHAR szVersion[NETDEV\_LEN\_256];  BYTE byRes[32];  }NETDEV\_CLIENT\_VERSION\_INFO\_S, \*LPNETDEV\_CLIENT\_VERSION\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 客户端类型参见枚举# [NETDEV\_CLIENT\_TYPE\_E](#_客户端类型) |
| szVersion | 客户端独立版本号，范围[0, 255] |
| byRes | 保留字段 |

**See also：**

[NETDEV\_CLIENT\_VERSION\_INFO\_LIST\_S](#_客户端版本信息列表)

### 设备信息

|  |
| --- |
| typedef struct tagNETDEVDeviceInfo  {  INT32 dwDevType;  INT16 wAlarmInPortNum;  INT16 wAlarmOutPortNum;  INT32 dwChannelNum;  BYTE byRes[48];  }NETDEV\_DEVICE\_INFO\_S,\*LPNETDEV\_DEVICE\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwDevType | 设备类型,参见枚举#[NETDEV\_DEVICE\_TYPE\_E](#_错误码列表) |
| wAlarmInPortNum | 报警输入个数 |
| wAlarmOutPortNum | 报警输出个数 |
| dwChannelNum | 通道个数 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_GetDeviceInfo](#_获取设备信息)

### 设备详细信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stDevBasicInfo | 设备基本信息 |
| stDevFirewareInfo | 设备固件信息 |

**See also：**

[NETDEV\_GetDeviceInfo\_V30](#_查询设备详细信息)

### 设备固件信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szDevModel | 设备型号 |
| szFireVersion | 软件版本号 |
| szSerialNum | 设备序列号 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_DEV\_INFO\_V30\_S](#_设备详细信息)

### 定位信息

|  |
| --- |
| typedef struct tagNETDEVGeolocationInfo  {  FLOAT fLongitude;  FLOAT fLatitude;  }NETDEV\_GEOLACATION\_INFO\_S, \*LPNETDEV\_GEOLACATION\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| fLongitude | 经度 |
| fLatitude | 纬度 |

**See also：**

[NETDEV\_GetGeolocationInfo](#_获取设备定位信息)

### WIFISnifferMac数组信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 数量 |
| astMACList | WIFISnifferMac信息 |

**See also：**

[NETDEV\_GetWifiSnifferMacList](#_获取wifi_sniffer_MAC地址信息)

### WIFISnifferMac信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| aszMACInfo | WiFi sniffer mac 地址 |
| wMaxSig | Mac地址对应最大信号强度,单位dBm |
| aszMaxSigTime | Mac地址对应最大信号强度获取时间 |

**See also：**

[NETDEV\_WIFISNIFFER\_MAC\_LIST\_S](#_WIFISnifferMac数组信息)

### 视频通道详细信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道ID |
| bPtzSupported | 是否支持云台 |
| enStatus | 通道状态 |
| dwStreamNum | 流个数，当enStatus为[NETDEV\_CHL\_STATUS\_UNBIND](#_通道状态) 时，此值无效 |
| szChnName | 通道名称 |
| byRes | 保留字段 |

**See also：**

NETDEV\_QueryVideoChlDetailList

### 视频通道详细信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道ID |
| bPtzSupported | 是否支持云台 |
| enStatus | 通道状态 |
| dwStreamNum | 流个数 |
| enChannelType | 通道类型 |
| enVideoFormat | 视频输入制式，当ChannelType为NETDEV\_CHL\_TYPE\_DIGITAL时，此值无效(note: 该字段仅对混合NVR有效) |
| enAddressType | IP地址类型 |
| szIPAddr | IP地址 |
| dwPort | 端口号 |
| szChnName | 通道名称 |
| allowDistribution | 是否允许流分发 |
| dwDeviceType | 通道接入的设备类型，参见枚举[NETDEV\_CHANNEL\_CAMERA\_TYPE\_E](#_通道下接入的设备类型(目前仅NVR使用)) |
| szManufacturer | 厂商，范围[0,31] |
| szDeviceModel | 设备型号，范围[0,31] |
| udwAccessProtocol | 接入协议类型,ONVIF协议:1 ,私有协议:2, 国标:3 ,自定义协议:使用自定义协议, 需要使用  [NETDEV\_GetDevConfig](#_获取设备的配置信息)/[NETDEV\_SetDevConfig](#_设置设备的配置信息)和获取和配置自定义协议配置信息 |
| pstExtendedInformation | 附加属性, 需要自行申请内存 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_QueryVideoChlDetailListEx](#_查询视频通道信息列表)

### 视频通道详细信息拓展

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwIsPoEPort | 是否支持PoE口, 0-否, 1-是 |
| udwPoEStatus | PoE口状态参见[NETDEV\_POE\_STATUS\_E](#_POE口状态类型枚举) |
| szAccessAddress | 设备访问地址, 范围[0, 63]NVR需要 |
| szMACAddress | 接入设备MAC地址, 长度范围[0, 63], NVR支持 |
| szDDNSAddress | DDNS服务器地址，范围[0,64] |
| udwOffReason | 接入设备离线原因参见[NETDEV\_CHN\_OFF\_REASON\_E](#_接入设备离线原因类型枚举) |
| udwRemoteIndex | 远程通道号(多通道设备的通道号) |
| szGBID | 国标资源编码，范围[0, 31] |
| udwAddType | 添加方式, 0-手动, 1-即插即用 |
| udwOrgID | 组织ID, 一体机需要 |
| udwDevID | 父设备ID, 一体机需要 |
| szDevName | 父设备名称，范围[0, 64] 一体机需要 |
| udwChlIndex | 通道索引ID, 一体机需要 |
| udwAudioResID | 对应的音频通道ID, 一体机需要 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_VIDEO\_CHL\_DETAIL\_INFO\_EX\_S](#_视频通道详细信息_1)

### 告警/事件上报信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwReportType | 上报类型，参见枚举#[NETDEV\_REPORT\_TYPE\_E](#_告警/事件上报类型) |
| stAlarmInfo | 告警信息，当dwReportType为[NETDEV\_REPORT\_TYPE\_ALARM](#_告警/事件上报类型)时有效 |
| stEventInfo | 事件信息，当dwReportType为[NETDEV\_REPORT\_TYPE\_EVENT](#_告警/事件上报类型)时有效 |

**See also：**

[NETDEV\_SetAlarmCallBack\_V30](#_注册回调函数,接收报警消息等)

### 告警消息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwAlarmType | 告警类型，参见枚举# [NETDEV\_ALARM\_TYPE\_E](#_告警类型枚举_1) |
| dwAlarmSubType | 告警子类型，参见[NETDEV\_ALARM\_SUBTYPE\_E](#_告警子类型枚举) |
| dwAlarmLevel | 告警等级，到级，级最严重 |
| tAlarmTimeStamp | 告警发生时间 |
| dwChannelID | 通道ID,非一体机设备使用 |
| dwAlarmID | 告警ID，一体机设备使用 |
| dwAlarmSrcID | 告警源ID 参见枚举# [NETDEV\_ALARM\_SRC\_TYPE\_E](#_告警资源类型) |
| szAlarmSrc | 告警源信息（名称） |
| IsAlarmSnapExisted | 告警是否有抓图0没有抓图1有抓图 vms使用 |
| wIndex | 索引号 |
| dwTotalBandWidth | 当前带宽总量,单位为MBps |
| dwUnusedBandwidth | 未使用的带宽,单位为MBps |
| dwTotalStreamNum | 总路数 |
| dwFreeStreamNum | 未使用路数 |
| dwMediaMode | 流类型,参见枚举#[NETDEV\_MEDIA\_MODE\_E](#_媒体类型枚举类型定义) |
| dwEventCode | 事件类型，用于上报解码层事件类型，参见枚举#  [NETDEV\_PLAYER\_RUN\_INFO\_TYPE\_E](#_解码层上报运行信息的类型的枚举定义) |
| dwReserved | 异常上报保留参数，用于上报解码层保留参数 |
| szFileName | ND上报字符串参数信息 |
| szDeviceID | 告警设备ID，国标协议接入时填写国标注册码。长度[1,32]。IPC、VM平台支持 |
| szRelatedID | 告警事件与告警数据的关联ID,同一个相机内全局唯一。长度为15个字符.当告警存在与之关联数据时，需携带此字段 |
| dwObjectNum | 目标个数 |
| pstObjectList | 目标列表，需根据dwObjectNum动态申请内存，参见 [NETDEV\_OBJECT\_LIST\_S](#_目标列表结构题) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_REPORT\_INFO\_S](#_告警/事件上报信息)

### 事件信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 资源数量 |
| astEventRes | 事件资源信息 |
| dwEventActionType | 事件类型，参见枚举# [NETDEV\_EVENT\_ACTION\_TYPE\_E](#_事件动作类型枚举) |
| pstEventRes | 超过NETDEV\_MAX\_EVENT\_RES\_SIZE的事件资源信息需要动态申请 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_REPORT\_INFO\_S](#_告警/事件上报信息)

### 事件资源信息

|  |
| --- |
| typedef struct tagNETDEVEventRes  {  INT32 dwResType;  INT32 dwResID;  INT32 dwFirstSubResID;  INT32 dwSecondSubResID;  BYTE byRes[16];  }NETDEV\_EVENT\_RES\_S, \*LPNETDEV\_EVENT\_RES\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwResType | 资源类型，参见枚举# [NETDEV\_EVENT\_RES\_TYPE\_E](#_事件资源类型枚举) |
| dwResID | 资源ID |
| dwFirstSubResID | 第一子资源ID |
| dwSecondSubResID | 第二子资源ID 不同资源类型对应意义不同。如：电视墙分屏资源的资源ID是电视墙ID，第一子资源ID是窗口ID，第二子资源ID就是分屏ID |
| byRes | 保留字段 |

**See also：**

[NETDEV\_EVENT\_INFO\_S](#_事件信息)

### 拉告警信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| enAlarm | 告警类型 |
| dwChancelId | 通道号,用于NVR |

**See also：**

NETDEV\_PullAlarm

### 客流量统计

|  |
| --- |
| typedef struct tagNETDEVTrafficStatisticsInfo  {  BOOL bIsSuccess; /\* 查询是否成功\*/  INT32 dwChannelID; /\* 通道号\*/  UINT32 udwSize; /\* 时间段个数\*/  UINT32 audwEnterCount[NETDEV\_PEOPLE\_CNT\_MAX\_NUM]; /\* 进入人数统计\*/  UINT32 audwExitCount[NETDEV\_PEOPLE\_CNT\_MAX\_NUM]; /\* 离开人数统计\*/  BYTE byRes[6]; /\* 保留字段 \*/  }NETDEV\_TRAFFIC\_STATISTICS\_INFO\_S, \*LPNETDEV\_TRAFFIC\_STATISTICS\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bIsSuccess | 查询是否成功 |
| dwChannelID | 通道号 |
| udwSize | 时间段个数 |
| audwEnterCount | 进入人数统计 |
| audwExitCount | 离开人数统计 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_FindNextTrafficStatisticInfo](#_逐个查询获取到的客流量信息)

### 客流量统计命令

|  |
| --- |
| typedef struct tagNETDEVTrafficStatisticsCond  {  INT32 dwChannelID;  INT32 dwStatisticsType;  INT32 dwFormType;  INT64 tBeginTime;  INT64 tEndTime;  }NETDEV\_TRAFFIC\_STATISTICS\_COND\_S, \*LPNETDEV\_TRAFFIC\_STATISTICS\_COND\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道号 |
| dwStatisticsType | 统计模式参考# [NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_E](#_客流量统计模式类型) |
| dwFormType | 统计报表参考# [NETDEV\_FORM\_TYPE\_E](#_客流量统计报表类型) |
| tBeginTime | 起始时间 |
| tEndTime | 结束时间 |

**See also：**

[NETDEV\_GetTrafficStatistic](#_获取客流量统计)、[NETDEV\_StartTrafficStatistic](#_开始客流量统计查询)

### 客流量统计

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 报表长度根据不同报表类型长度不同 |
| adwEnterCount | 进入人数计数值 |
| adwExitCount | 离开人数计数值 |

**See also：**

[NETDEV\_GetTrafficStatistic](#_获取客流量统计)

### 客流量上报时间间隔

|  |
| --- |
| typedef struct tagNETDEVTrafficStatisticsReportTime  {  UINT32 udwReportTimeInterval;  BOOL bEnable;  }NETDEV\_TRAFFIC\_STATISTICS\_REPORT\_TIME\_S, \*LPNETDEV\_TRAFFIC\_STATISTICS\_REPORT\_TIME\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwReportTimeInterval | 时间间隔单位秒 |
| bEnable | 使能，0：未开启，1开启 |

**See also：**

[NETDEV\_GetTrafficStatisticReportTime](#_获取客流量上报时间间隔)

### 多通道客流量统计命令

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stChannelIDs | 通道ID列表 |
| udwStatisticsType | 参见[NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_E](#_客流量统计模式类型) |
| udwFormType | 参见[NETDEV\_TRAFFIC\_STATIC\_FORM\_TYPE\_E](#_客流量报表类型) |
| tBeginTime | 起始时间 |
| tEndTime | 结束时间 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_StartMultiTrafficStatistic](#_多通道开始客流量统计查询)

### 客流量统计信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道号 |
| tReportTime | 上报时间（unix时间戳） |
| tInterval | 间隔时间 |
| dwEnterNum | 进入人数 |
| dwExitNum | 离开人数 |
| dwTotalEnterNum | 所有进入人数 |
| dwTotalExitNum | 所有离开人数 |

**See also：**

[NETDEV\_PassengerFlowStatisticCallBack\_PF](#_注册客流量统计上报回调函数)

### XW设备能力集信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stTVWall | 电视墙能力集 |
| stVideoFormats | 视频输出制式列表 |
| stWnd | 窗口能力集 |
| stBaseMap | 底图能力集 |
| stVirtualLED | 虚拟LED能力集 |
| stSequence | 轮巡能力集 |
| stSupport | 特性能力集 |
| bIsRSASupported | 是否使用RSA加密 |
| bIsDevMgrSupported | 是否支持管理设备 |
| bIsDecodeSupported | 是否支持解码 |
| bIsChromaCalSupported | 是否支持色度校正 |
| udwSuggestVideoFormat | 推荐分辨率 参见#[NETDEV\_VIDEO\_FORMAT\_E](#_分辨率枚举) |
| udwSupportExVideoFormatNum | 支持的扩展视频输出制式数量 |
| pstVideoFormatsEx | 支持的扩展视频输出制式列表 ,需根据udwSupportExVideoFormatNum动态分配内存 |
| udwBatchCapabilities | 批量操作能力，按Bit位描述 Bit1:开关窗 Bit2:漫游 Bit3:分屏 Bit4:窗口序号 |
| bIsOrderNOSupported | 是否支持控制序号 |
| pstAlarmCap | 设备告警能力 |
| udwDeviceID | 设备ID，仅VMS支持 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_XW\_GetCapability](#_获取能力集)

### 电视墙能力集结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNumMax | 电视墙最大数量 |
| udwSceneMax | 单个电视墙的最大场景数 |
| udwRowMax | 单个电视墙的最大行数 |
| udwColumnMax | 单个电视墙的最大列数 |
| udwScenesPlanMax | 单个电视墙最大场景组计划数 |
| udwMaxPerTVWallVideoOutNum | 单电视墙可以绑定的最大视频输出口数量，电视墙行列乘积不能超过该限定值 |
| byRes | 保留字段 |

### 视频输出制式列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSize | 支持的视频输出制式数量 |
| adwFormatList | 支持的视频输出制式 [NETDEV\_VIDEO\_FORMAT\_E](#_分辨率枚举) |
| byRes | 保留字段 |

### 窗口能力集结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwWndMaxPerScene | 单场景的最大窗口数量 |
| udwAreaMax | 单物理输出口的窗口面积 |
| udwPaneMax | 单窗口的最大分屏数量 |
| stResolutionMin | 最小窗口的像素 |
| byRes | 保留字段 |

### 底图能力集结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNumMax | 设备支持的最大底图数量 |
| udwNumMaxPerScene | 单场景能绑定的底图数量 |
| udwMaxBaseMapSize | 支持图片的大小 单位：MB |
| stResolutionMax | 底图的最大分辨率 |
| stResolutionMin | 底图的最小分辨率 |
| szBaseMapType | 支持图片的格式 |
| byRes | 保留字段 |

### 虚拟LED能力集结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNumMaxPerScene | 单场景最大虚拟LED数量 |
| udwRowMax | 单虚拟LED最大行数 |
| udwCharMaxPerRow | 单行文字的最大个数 |
| udwRollingNumPerSceneMax | 滚动虚拟LED最大个数 |
| udwTextHightMax | 字高最大值 |
| udwGapMax | 自适应间隔最大像素 |
| udwCharsMax | 最大字符个数限制 |
| udwSupVirtualLEDDifColors | 支持虚拟LED显示不同的颜色 0：不支持 1：支持 |
| udwMaxPerSceneTimeLEDNum | 单场景能设置的最大时间虚拟LED数量 |
| byRes | 保留字段 |

### 轮巡能力集结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSrcNumMax | 单个轮训最大资源数量 |
| udwPaneNumMax | 单分屏轮巡最大数量 |
| udwWndNumMax | 单窗口轮巡最大数量 |
| udwSceneNumMax | 场景轮巡最大个数 |
| udwVideoSourceMax | 单个轮巡资源最大监控点数 |
| udwSeqResourceInSceneMax | 单个场景下最大的轮巡资源数 |
| byRes | 保留字段 |

### 特性能力集结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPane | 分屏能力 [NETDEV\_PANE\_CAP\_E](#_分屏能力枚举) |
| udwWndRoaming | 窗口漫游能力 [NETDEV\_ROAMING\_CAP\_E](#_窗口漫游能力枚举) |
| udwWnd | 开窗能力 [NETDEV\_XW\_WND\_CAP\_E](#_开窗能力枚举) |
| bBackToShow | 回显能力 0:不支持 1:支持 |
| bWndOverlay | 窗口叠加能力 0:不支持 1:支持 |
| bTopWndSetTransp | 顶层窗口设置透明度 0:不支持 1:支持 |
| udwMaxWinNumPerScreen | 单个物理输出口能支持的最大窗口数量 |
| stPaneType | 支持的分屏 |
| udwMaxEncodeDevicesNum | 接入编码设备的最大数量 |
| udwMaxVideoSourceNum | 接入通道的最大数量 |
| udwSupportWindowTypeNum | 支持的开窗模式数量 |
| paudwWindowType | 具体的开窗模式枚举 ，参考枚举定义#[NETDEV\_XW\_LAYOUT\_NUM\_E](#_屏幕输出制式枚举) ,需根据支持的开窗模式数量动态分配内存 |
| bAutoSelectSignalSrc | 自动选择信号源 0:不支持 1:支持 |
| bSupportCustomizeOutputFormat | 是否支持自定义输出制式 0:不支持 1:支持 |
| bSupportCustomizeInputFormat | 是否支持本地输入自定义EDID, 0:不支持，1：支持 |
| byRes | 保留字段 |

### XW设备支持的分屏信息结构体

|  |
| --- |
| typedef struct tagNETDEVXWPaneType  {  UINT32 udwSize;  UINT32 \*pudwPaneType;  }NETDEV\_XW\_PANE\_TYPE\_S, \*LPNETDEV\_XW\_PANE\_TYPE\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSize | 支持的分屏数量 |
| pudwPaneType | 支持的分屏类型 [NETDEV\_XW\_LAYOUT\_NUM\_E](#_屏幕输出制式枚举) |

### "XW"设备扩展视频输出制式能力集结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwVideoOutID | 视频输出通道ID,是视频通道输出口的唯一标识。通过获取设备物理资源接口获取，和创建电视墙绑定通道接口下发的VideoOutID保持一致 |
| udwFormatNum | 视频输出通道支持的视频制式数量 |
| adwFormatList | 支持的视频输出制式 [NETDEV\_VIDEO\_FORMAT\_E](#_分辨率枚举) |
| byRes | 保留字段 |

### 通道数量信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwVideoInNum | 视频输入通道数量 |
| udwVideoOutNum | 视频输出通道数量 |
| udwAudioInNum | 音频输入通道数量 |
| udwAudioOutNum | 音频输出通道数量 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_XW\_GetChannelsNum](#_获取本地编码通道数量)

### 解码器通道信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwVideoInNum | 视频输入通道数量 需要给出ADU/CDU支持的最大通道数 |
| pstVideoIn | 视频输入通道信息,需根据udwVideoInNum动态分配内存 |
| udwVideoOutNum | 视频输出通道数量 需要给出ADU/CDU支持的最大通道数 |
| pstVideoOut | 视频输出通道信息,需根据udwVideoOutNum动态分配内存 |
| udwAudioInNum | 音频输入通道数量 需要给出ADU/CDU支持的最大通道数 |
| pstAudioIn | 音频输入通道信息,需根据udwAudioInNum动态分配内存 |
| udwAudioOutNum | 音频输出通道数量 需要给出ADU/CDU支持的最大通道数 |
| pstAudioOut | 音频输出通道信息,需根据udwAudioOutNum动态分配内存 |
| udwSerialNum | 串口通道的数量 需要给出ADU/CDU支持的最大通道数 |
| pstSerialInfo | 串口通道信息,需根据udwSerialNum动态分配内存 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_XW\_GetChannelsInfoList](#_获取视频通道信息列表)

### 视频输入通道信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道ID |
| dwIndex | 序号 |
| udwVideoPortType | 通道类型 [NETDEV\_VIDEO\_PORT\_TYPE\_E](#_视频端口枚举) |
| udwStatus | 通道在线状态 参见 [NETDEV\_CHN\_STATUS\_E](#_编码通道状态) |
| byRes | 保留字段 |

### XW视频输出通道信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道ID |
| dwSlotID | 槽位ID |
| dwIndex | 序号 |
| udwVideoPortType | 通道类型 [NETDEV\_VIDEO\_PORT\_TYPE\_E](#_视频端口枚举) |
| udwStatus | 通道在线状态 参见 [NETDEV\_CHN\_STATUS\_E](#_编码通道状态) |

### 音频输入通道信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道ID |
| udwAudioPortType | 通道类型 [NETDEV\_AUDIO\_PORT\_TYPE\_E](#_音频通道端口枚举) |
| udwIndex | 序号，和设备丝印的序号保持一致 |
| udwStatus | 通道在线状态 参见 [NETDEV\_CHN\_STATUS\_E](#_编码通道状态) |
| byRes | 保留字段 |

### 音频输出通道信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道ID |
| udwAudioPortType | 通道类型 [NETDEV\_AUDIO\_PORT\_TYPE\_E](#_音频通道端口枚举) |
| udwIndex | 序号，和设备丝印的序号保持一致 |
| udwStatus | 通道在线状态 参见 [NETDEV\_CHN\_STATUS\_E](#_编码通道状态) |
| byRes | 保留字段 |

### 串口通道信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道ID |
| dwIndex | 序号 |
| udwSerialPortType | 通道类型 [NETDEV\_XW\_SERIAL\_TYPE\_E](#_串口类型枚举) |
| udwStatus | 通道在线状态 参见 [NETDEV\_CHN\_STATUS\_E](#_编码通道状态) |
| byRes | 保留字段 |

### 电视墙信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTVWallID | 电视墙ID |
| szTVWallCode | 电视墙编码（可选字段），由设备端分配，字符串长度范围[1,64] |
| szName | 电视墙名称 |
| udwOrderNo | 电视墙序号 |
| stScreenInfo | 电视墙屏幕配置 |
| udwFormatSpecNum | 特殊输出制式数量 |
| astFormatSpec | 特殊输出制式配置 |
| udwLedSpecNum | 特殊模组框数量 |
| astLedSpec | 特殊模组框配置 |
| udwVideoOutNum | 物理输出端口数量 |
| astVideoOut | 物理输出端口列表 |
| udwTVWallType | 电视墙类型 |
| dwXWDeviceID | -1表示DC设备，>0表示ADU设备ID |
| udwZoomEnabled | 是否缩放，0:不缩放，1:缩放 |
| byRes | 保留字段 |

**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)

### 电视墙屏幕配置结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwRowNum | 电视墙行单元数量 |
| udwColNum | 电视墙列单元数量 |
| udwFormat | 屏幕输出制式 [NETDEV\_VIDEO\_FORMAT\_E](#_分辨率枚举) |
| szFormat | 无需配置 屏幕输出制式 |
| stResolution | 屏幕像素 |
| byRes | 保留字段 |

### 特殊输出制式配置结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwScreenID | 屏幕序号(从1开始) |
| udwFormat | 屏幕输出制式 [NETDEV\_VIDEO\_FORMAT\_E](#_分辨率枚举) |
| szFormat | 无需配置 屏幕输出制式 |
| byRes | 保留字段 |

### 特殊模组框配置结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwScreenID | 屏幕序号(从1开始) |
| stResolution | 屏幕像素 |
| byRes | 保留字段 |

### 物理输出端口配置结构体

|  |
| --- |
| typedef struct tagNETDEVTVWallVOUTList  {  UINT32 udwScreenID;  UINT32 udwVideoOutID;  BYTE byRes[32];  }NETDEV\_TVWALL\_VIDED\_OUT\_S, \*LPNETDEV\_TVWALL\_VIDED\_OUT\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwScreenID | 屏幕序号(从1开始) |
| udwVideoOutID | 物理输出端口ID |
| byRes | 保留字段 |

### 虚拟LED信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwLEDID | 虚拟LED的ID |
| udwModifyCmd | 修改可选命令字 [NETDEV\_XW\_LED\_CMD\_E](#_虚拟LED修改可选命令字枚举) |
| udwEnable | 虚拟LED是否启用 0不使能 1使能 |
| udwType | 虚拟LED类型 [NETDEV\_XW\_LED\_TYPE\_E](#_虚拟LED类型枚举) |
| stArea | 虚拟LED的位置信息 |
| stBackground | 背景 |
| stFontInfo | 文字信息描述 |
| szText | 文字内容 |
| byRes | 保留字段 |

**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)

### 背景结构体

|  |
| --- |
| typedef struct tagNETDEVXWBackground  {  UINT32 udwTransparency;  UINT32 udwBaseColor;  BYTE byRes[16];  }NETDEV\_XW\_BACKGROUND\_S, \*LPNETDEV\_XW\_BACKGROUND\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTransparency | 透明度 0－全透明 默认,100－不透明 |
| udwBaseColor | 颜色 RGB方式:0xABCDEF AB表示R 00~FF;CD表示G 00~FF;EF表示B 00~FF;默认：0xFFFFFF（白色） |
| byRes | 保留字段 |

### 文字信息描述结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwFont | 字体 [NETDEV\_XW\_LED\_FONT\_FORMAT\_E](#_虚拟LED字体枚举) |
| udwSize | 字体大小 0到7参照枚举：[NETDEV\_XW\_LED\_FONT\_SIZE\_E](#_虚拟LED字体大小枚举),其余取值范围[48,1024]，表示实际像素高度 |
| udwReferFontSize | 参考字体大小 当udwSize选择自适应后该字段必选 值范围参照Size字段 |
| udwGap | 字间距 参见枚举 [NETDEV\_XW\_LED\_FONT\_GAP\_E](#_虚拟LED字间距枚举) |
| udwHorizontalAlign | 水平对齐方式 [NETDEV\_XW\_HORIZONTAL\_ALIGN\_MODE\_E](#_文字水平对齐方式枚举) |
| udwVerticalAlign | 垂直对齐方式 [NETDEV\_XW\_VERTICAL\_ALIGN\_MODE\_E](#_文字垂直对齐方式枚举) |
| udwColor | 颜色 RGB方式:0xABCDEF AB表示R 00~FF;CD表示G 00~FF;EF表示B 00~FF;默认：0xFFFFFF（白色） |
| udwScrollMode | 文字滚动方式 [NETDEV\_XW\_LED\_SCROLLING\_MODE\_E](#_文字滚动方式枚举) |
| udwScrollRateSec | 滚动速度，单位：秒,有条件选择，当ScrollingMode为非1值时，必选；默认1秒 1x－10x可选，1x最慢 ，当文字滚动类型为2/3/4/5时必选 |
| udwReferFontGap | 参考字间距大小,当udwGap选择自适应时该字段有效，取值范围0~1024（实际像素宽度） |
| bVoiceBroadcast | 文字是否进行播报 0：否 1：是 |
| bBold | 粗体 0:不加粗, 1:加粗 |
| bSlope | 倾斜 0:不倾斜, 1:倾斜 |
| bUnderline | 下划线 0:无下划线, 1:有下划线 |
| udwLineSpace | 指定行间距像素 |
| byRes | 保留字段 |

### 窗口信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwWndID | 窗口ID |
| udwModifyCmd | 修改可选命令字 [NETDEV\_XW\_WND\_CMD](#_场景窗口修改可选命令字枚举) |
| szWndName | 窗口名称 |
| udwPaneMod | 分屏模式,默认1分屏 [NETDEV\_XW\_LAYOUT\_NUM\_E](#_屏幕输出制式枚举) |
| udwLayer | 图层，数字大的在最上层 |
| udwTransparency | 透明度：默认0 不透明,100 全透明 0-100的其他值表明部分透明 |
| stArea | 位置信息 |
| udwZoomType | 放大类型 参见[NETDEV\_XW\_ZOOM\_TYPE\_E](#_窗口放大类型枚举) |
| udwSplitIndex | 分屏序号 当udwZoomType为NETDEV\_XW\_ZOOM\_TYPE\_SPLIT时有效 |
| udwSchemeResID | 轮巡资源ID |
| udwSeqStatus | 轮巡状态参见[NETDEV\_SEQ\_STATUS\_E](#_轮巡状态枚举_1) |
| udwPaneSize | 分屏数量 |
| astPaneInfoList | 所有分屏信息 |
| udwWndType | 窗口类型 参见枚举[NETDEV\_TVWALL\_WND\_TPYE](#_窗口类型枚举) |
| udwDChlID | DC通道ID |
| udwStatus | 0表示不锁定，1表示锁定 |
| udwScaleEnabled | 缩放状态 详见 [NETDEV\_SCALE\_ENABLED\_E](#_缩放状态枚举) 该字段仅在LED发送控制设备使用，默认为满比例缩放 |
| byRes | 保留字段 |

**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)

### 窗口分屏信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPaneID | 分屏序号 |
| dwChannelID | 通道ID |
| udwStreamIndex | 流索引 |
| udwTaskNo | 业务号 |
| udwTransProtocal | 传输协议 参见 [NETDEV\_PROTOCAL\_E](#_媒体传输协议枚举) |
| udwSessionType | 分屏业务类型 参见 [NETDEV\_SESSION\_TYPE\_E](#_分屏业务类型枚举) |

### 窗口列表信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwOperation | 窗口批量操作类型 0：窗口批量漫游 1：窗口批量设置分屏 |
| udwNum | 窗口数量 |
| pstSceneWinInfo | 窗口详细信息列表 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_XW\_CreateBatchWnd](#_批量开窗)、[NETDEV\_XW\_ModifyBatchWnd](#_批量修改窗口)

### 批量开窗场景窗口信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwReqSeq | 请求数据序号 |
| stSenceWndInfo | 窗口信息 |
| byRes | 保留字段 |

### 批量操作结果信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSize | 窗口数量 |
| udwLastChange | 摘要字 |
| pstResultInfo | 窗口信息,根据窗口数量动态申请内存 |
| byRes | 保留字段 |

**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)

### 批量开窗场景窗口返回信息结构体

|  |
| --- |
| typedef struct tagNETDEVXWSenceBatchResultWnd  {  UINT32 udwReqSeq;  UINT32 udwResuleCode;  UINT32 udwWinID;  }NETDEV\_XW\_BATCH\_RESULT\_WND\_S, \*LPNETDEV\_XW\_BATCH\_RESULT\_WND\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwReqSeq | 请求数据序号 |
| udwResuleCode | 返回错误码 |
| udwWinID | 窗口ID |

### 屏幕缩放信息结构体

|  |
| --- |
| typedef struct tagNETDEVXWScreenZoomInfo  {  UINT32 uOperatedType;  UINT32 uOperatedMode;  UINT32 uSplitScreenIndex;  }NETDEV\_XW\_SCREEN\_ZOOM\_INFO\_S, \*LPNETDEV\_XW\_SCREEN\_ZOOM\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| uOperatedType | 屏幕操作类型,参见[NETDEV\_XW\_SCREEN\_ZOOM\_TYPE\_E](#_屏幕放大类型枚举) |
| uOperatedMode | 屏幕操作模式,参见[NETDEV\_XW\_SCREEN\_ZOOM\_MODE\_E](#_屏幕放大模式枚举) |
| uSplitScreenIndex | 需要放大到全屏的分屏序号,当uOperatedType＝0 & uOperatedMode=1时有效 |

**See also：**

[NETDEV\_XW\_ZoomScreen](#_电视墙窗口放大/还原)

### 解码设备ID信息结构体

|  |
| --- |
| typedef struct tagNETDEVXWDisplayerID  {  UINT32 udwTVWallID;  UINT32 udwWndID;  UINT32 udwPaneID;  BYTE byRes[16];  }NETDEV\_XW\_DISPLAYER\_ID\_S, \*LPNETDEV\_XW\_DISPLAYER\_ID\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTVWallID | 电视墙ID |
| udwWndID | 窗口ID |
| udwPaneID | 分屏序号 |
| byRes | 保留字段 |

**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)

### 视频源信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stDisplayerID | 设备ID信息 |
| udwVideoInID | 视频源ID |
| szRealplayURL | 媒体流URL 本地输入时无需配置 |
| udwStreamID | 媒体流序号 |
| stAuthentication | 鉴权信息 |
| udwTransProtocol | 传输协议 [NETDEV\_LAPI\_TRANS\_PROTOCOL\_E](#_媒体传输协议枚举_2) 本地输入时无需配置 |
| udwMulticast | 单组播 [NETDEV\_STREAM\_TYPE\_E](#_流类型枚举) 本地输入时无需配置 |
| udwType | 实况类型 参考 [NETDEV\_XW\_REALPLAY\_TYPE\_E](#_实况业务类型枚举) |
| udwDecodeType | 解码类型 参考[NETDEV\_XW\_DECODE\_TYPE\_E](#_解码类型枚举) |
| udwManageType | 资源管理类型，参见枚举[NETDEV\_XW\_MANAGE\_TYPE\_E](#_资源管理类型枚举) |
| pstContentInfo | 文字信息 |
| byRes | 保留字段 |

**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)

### 鉴权信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bIsNeedAuthentication | 是否需要鉴权 |
| szUserName | 视频源用户名 |
| szPassword | 视频源密码 |
| enAuthModel | 保留字段，暂不使用。加密类型 参考枚举#[NETDEV\_AUTH\_MODEL\_E](#_加密类型枚举) |
| byRes | 保留字段 |

### 文字信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szContent | 文字内容 |
| udwFontType | 文字字体 参见枚举定义# [NETDEV\_XW\_LED\_FONT\_FORMAT\_E](#_虚拟LED字体枚举) |
| udwFontSize | 文字大小 0：自适应，48-1024:指定的字体像素高度 |
| udwColor | 文字颜色，颜色以RGB的方式描述，0xABCDEF，AB：表示R的值，00~FF，CD：表示G的值，00~FF，EF：表示B的值，00~FF。默认：0xFF0000（红色） |
| udwPosition | 文字位置 参加枚举定义# [NETDEV\_XW\_CONTENT\_POSITION\_E](#_文字位置枚举) |
| byRes | 保留字段 |

### 被动解码发送控制参数信息结构体

|  |
| --- |
| typedef struct tagNETDEVPassiveDecodeSend  {  INT32 dwTransType;  INT32 dwBufSize;  CHAR \*pszBuffer;  BYTE byRes[128];  }NETDEV\_PASSIVE\_SEND\_S, \*LPNETDEV\_PASSIVE\_SEND\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwTransType | 传输方式 |
| dwBufSize | 数据大小 |
| pszBuffer | 数据内容 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SendPassiveData](#_向被动解码通道发送数据)

### 流业务的详细信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTaskNo | 业务号 |
| udwType | 业务类型 [NETDEV\_XW\_VIDEO\_TYPE\_E](#_流业务类型枚举) |
| stStreamInfoSrc | 源端媒体流信息 |
| stStreamInfoDest | 目的端媒体流信息 |
| stStreamInfoMsg | 媒体流具体信息 |

**See also：**

[NETDEV\_XW\_FindNextStreamInfo](#_逐个获取业务流信息)

### 轮巡资源基本信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 轮巡资源的唯一ID |
| szName | 轮巡资源名称 |
| szDescription | 轮巡资源描述信息 |
| bEnabled | 轮巡资源是否启用 |
| udwOrderNO | 轮巡资源资源序号,查询时必选 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_XW\_GetSequenceResList](#_获取设备全部轮巡资源)

### 轮巡资源详细信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stSeqResBaseInfo | 轮巡资源基本信息 |
| udwIntervalTime | 轮巡间隔时间，单位：秒 |
| udwVideoSrcNum | 组轮巡视频源个数 |
| pstVideoInList | 组轮巡视频源列表信息,内存由调用者维护, 规格参照能力集接口获取 |
| udwManageType | 资源管理类型，参见枚举[NETDEV\_XW\_MANAGE\_TYPE\_E](#_资源管理类型枚举) |
| byRes | 保留字段 |

**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)

### 轮巡视频源信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwVideoSrcIndex | 视频源的序号 |
| stVideoSrcBaseInfo | 视频源基本信息 |
| udwPresetIndex | 云台预置位序号，只有是轮巡时才需要配置 |
| byRes | 保留字段 |

### 远端实况的视频源基本信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwVideoInID | 视频源ID |
| szRealplayURL | 媒体流URL 本地输入时无需配置 |
| udwStreamID | 媒体流序号 |
| stAuthentication | 鉴权信息 |
| udwTransProtocol | 传输协议 [NETDEV\_LAPI\_TRANS\_PROTOCOL\_E](#_媒体传输协议枚举_2) 本地输入时无需配置 |
| udwMulticast | 单组播 [NETDEV\_STREAM\_TYPE\_E](#_流类型枚举) 本地输入时无需配置 |
| udwType | 实况类型 参考 [NETDEV\_XW\_REALPLAY\_TYPE\_E](#_实况业务类型枚举) |
| udwManageType | 资源管理类型，参见枚举[NETDEV\_XW\_MANAGE\_TYPE\_E](#_资源管理类型枚举) |
| byRes | 保留字段 |

### 场景基本信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSceneID | 场景ID |
| szName | 场景名称 |
| udwOrderNo | 场景序号 |
| udwTVWallID | 电视墙ID |
| udwStatus | 0表示不启用，1表示启用 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_XW\_CreateSceneInfo](#_创建单个场景信息)

### 场景修改信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stSceneBaseInfo | 场景信息 |
| dwModifyType | 修改类型，参见[NETDEV\_XW\_SCENE\_MODIFY\_TYPE\_E](#_场景修改类型枚举) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_XW\_ModifySceneInfo](#_修改场景信息)

### 场景详细信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szName | 场景名称 |
| udwTVWallID | 电视墙ID |
| szTVWallCode | 电视墙编码（可选字段），由设备端分配，字符串长度范围[1,64] |
| stBaseMapInfo | 底图信息 |
| udwVirtualLedNum | 虚拟LED数量 |
| pstVirtualLedInfo | 虚拟LED信息 |
| udwWndNum | 窗口数量 |
| pstWndInfo | 窗口信息 |
| udwLiveBindNum | 绑定实况业务数量 |
| pstVideoSource | 绑定实况业务信息 |
| udwSequenceBindNum | 绑定轮巡业务数量 |
| pstSequenceSourceList | 绑定轮巡业务信息 |
| udwWindowsSequenceBindNum | 绑定窗口组业务数量 |
| pstWindowsSequenceBindList | 绑定窗口组轮巡业务的详细信息 |
| udwLinkedSceneID | 关联场景ID |
| udwLinkedSceneOrderNo | 关联场景序号 |
| udwSceneSessionMode | 场景的业务模式 参见枚举定义# [NETDEV\_SCENE\_SESSION\_MODE\_E](#_场景业务模式枚举) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_XW\_GetCurrentSceneInfo](#_获取当前场景配置内容)、[NETDEV\_XW\_GetSceneInfo](#_获取场景信息)

### 所有底图信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 底图ID，整个设备唯一 |
| szName | 底图名称 |
| udwFormat | 底图格式 [NETDEV\_XW\_BASE\_TYPE\_E](#_底图类型枚举) |
| stResolution | 底图分辨率 |
| udwThumbnailData | 缩略图数据 |
| byRes | 保留字段 |

### 轮询资源列表信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwWndID | 窗口ID |
| udwSourceSize | 摄像机业务中视频源数量 |
| astSequenceSource | 摄像机业务中视频源信息 |
| byRes | 保留字段 |

### 轮询资源信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwInterval | 轮巡间隔 单位秒 |
| udwPreset | 预置位序号 |
| stVideoSourceBase | 摄像机业务中视频源信息 |
| byRes | 保留字段 |

### 场景简要信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stSceneBase | 场景基本 |
| udwIsCurrent | 是否当前场景 0:非当前场景 1:当前场景 |
| byRes | 保留字段 |

**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)

### 轮巡资源绑定信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 轮巡资源绑定信息ID,添加时可选 |
| udwSequenceResID | 轮巡资源ID |
| udwIntervalTime | 业务的时间间隔，单位：秒 |
| udwStatus | 轮巡状态，参见枚举[NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_E](#_轮巡状态枚举) |
| udwWndNum | 窗口数量 |
| audwWndIDList | 窗口ID列表，内存由用户维护，规格由能力集接口获取 |
| byRes | 保留字段 |

**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)

### 轮巡播放控制信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSequenceResBindID | 轮巡资源绑定信息ID |
| udwOperation | 操作命令类型 参见枚举[NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_E](#_轮巡播放控制枚举) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_XW\_SequencePlayControl](#_轮巡播放控制)

### 场景轮巡计划基本信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 计划ID |
| szName | 计划名称 |
| udwOrderNo | 计划序号 |
| szDescription | 计划描述信息 |
| udwStatus | 计划状态 参见[NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_E](#_轮巡状态枚举) |
| udwMode | 计划方式 参见[NETDEV\_XW\_SCENES\_PLAN\_MODE\_E](#_场景轮巡计划模式枚举) |
| stTimerInfo | 定时切换模式信息 |
| stTimeTableInfo | 时间表模式切换信息 |
| byRes | 保留字段 |

**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)

### 场景计划定时切换模式信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwIntervalTime | 时间间隔 单位：秒 |
| udwSceneNum | 场景个数 |
| pudwSceneIDList | 场景ID列表，内存由用户维护 |
| byRes | 保留字段 |

### 场景计划时间表模式信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTimePeriodNum | 时间分段个数 |
| pstTimeTableList | 计划时间表信息，内存由用户维护，大小由udwTimePeriodNum指定 |
| byRes | 保留字段 |

### 24小时时间表配置信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSceneID | 场景ID |
| szStartTime | 启动时间 hh:mm:ss |
| byRes | 保留字段 |

### 场景轮巡计划播放控制信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwScenesSeqPlanID | 场景轮巡计划ID |
| udwOperation | 操作命令类型 参见枚举[NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_E](#_轮巡播放控制枚举) |
| byRes | 保留字段 |

**See also：**

[NETDEV\_XW\_ScenesSeqPlanPlayControl](#_场景轮巡计划播放控制)

### 源端媒体流信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwVideoInChlID | 视频源ID |
| udwStreamID | 媒体流序号 |
| szIPAddr | 视频源IP地址 |
| udwPort | 端口号 |
| byRes | 保留字段 |

### 目的端媒体流信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwWndID | 窗口ID |
| udwPaneID | 分屏序号 |
| szIPAddr | 目的端IP地址 |
| udwPort | 端口号 |
| byRes | 保留字段 |

### 媒体流具体信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTransProtocol | 传输协议 [NETDEV\_LAPI\_TRANS\_PROTOCOL\_E](#_媒体传输协议枚举_2) |
| udwIsMulticast | 单组播 [NETDEV\_STREAM\_TYPE\_E](#_流类型枚举) |
| stResolution | 分辨率 |
| udwFrame | 帧率 |
| udwEncodeFormat | 编码格式 [NETDEV\_XW\_VIDEOENCODING\_E](#_编码格式枚举) |
| udwIsInvalidAfterReboot | 是否需要恢复 0 "XW"设备不负责恢复，1 "XW"设备恢复 |
| byRes | 保留字段 |

### 手动联动数据结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 联动类型 详见 [NETDEV\_LINKAGE\_TYPE\_E](#_手动联动类型枚举) |
| stPoint | 点坐标 Type为0和2时生效 |
| stRectangle | 矩形坐标 Type为1时生效 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_ManualLink](#_手动联动操作)

### 矩形坐标结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stTopLeft | 左上角的坐标点 |
| stBottomRight | 右下角的坐标点 |
| byRes | 保留字段 |

### 车牌识别报文数据信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPicNum | 图片个数 |
| stTmsXmlInfo | XML信息 |
| stTmsPicInfo | 图片信息 |

**See also：**

[NETDEV\_CarPlateCallBack\_PF](#_注册车牌识别上报回调函数)

### 车牌识别报文XML解析信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szCamID | 卡口相机编号 |
| szRecordID | 记录ID号 |
| szTollgateID | 卡口编号 |
| szPassTime | 经过时刻 |
| szLaneID | 车道编号 |
| szCarPlate | 车牌号码 |
| szIPAddr | 设备IP地址 |
| dwCarPlateColor | 号牌颜色，参见 [NETDEV\_TMS\_CAR\_PLATE\_COLOR\_E](#_号牌颜色类型枚举) |
| bRes | 保留字段 |

### 车牌识别报文图片解析信息结构体

|  |
| --- |
| typedef struct tagNETDEVTMSCarPlatePicInfo  {  UINT32 udwPicSize;  CHAR \*pcPicData;  }NETDEV\_TMS\_CAR\_PLATE\_PIC\_INFO\_S,\*LPNETDEV\_TMS\_CAR\_PLATE\_PIC\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPicSize | 图片大小 |
| pcPicData | 图片数据 |

### 人脸识别记录结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwRecordID | 人脸识别记录ID |
| udwRecordType | 人脸识别记录类型 参见# [NETDEV\_FACE\_PASS\_RECORD\_TYPE\_E](#_人脸通行记录类型枚举) |
| udwPassTime | 过人时间 UTC格式 |
| udwEventType | 事件类型 按BIT位进行类型描述，相应BIT为1，表示属于该类型，为0表示不属于该类型。BIT0:人脸抓拍BIT1:人脸匹配告警BIT2:人脸不匹配告警 |
| udwChannelID | 通道ID |
| udwMonitorRuleID | 告警对应的布控ID |
| szChannelName | 抓拍通道名称 |
| stCompareInfo | 比对信息 |
| byRes | 保留字段 |

**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)

### 人脸抓拍告警记录比对信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSimilarity | 相似度 |
| stMemberInfo | 人脸库成员信息 |
| stSnapshotImage | 抓拍图片 |
| pstPersonInfo | 人脸库成员信息(NVR支持) 查询匹配成功/失败记录携带，需动态申请内存 |
| pstFaceAttr | 人脸属性信息，需动态申请内存 |
| pstPersonAttr | 关联人员属性信息，需动态申请内存 |
| byRes | 保留字段 |

### 人脸库成员信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwReqSeq | 请求数据序号 |
| udwMemberID | 人脸库成员ID 只读 post消息时不带 |
| szMemberName | 成员名称 |
| udwMemberGender | 成员性别 参见枚举 [NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举) |
| szMemberBirthday | 成员出生日期 |
| stMemberRegionInfo | 成员地区信息 |
| stMemberIDInfo | 成员证件信息 |
| stMemberImageInfo | 人脸图片信息 |
| stMemberSemiInfo | 人脸半结构化信息 |
| udwCustomNum | 自定义属性值数量 |
| stCustomValue | 自定义属性值列表 |
| bIsMonitored | 是否已布控 0未布控，1已布控 |
| udwDBNum | 所属人脸库数量 |
| audwDBIDList | 人脸库ID列表 |
| byRes | 保留字段 |

### 抓拍图片信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stBigImage | 大图 |
| stSmallImage | 小图 |
| stArea | 区域坐标 |
| byRes | 保留字段 |

### 人脸库成员地区信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szNation | 国籍 |
| szProvince | 省份 |
| szCity | 城市 |
| byRes | 保留字段 |

### 成员证件信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 证件类型 参见枚举 [NETDEV\_FACE\_MEMBER\_ID\_TYPE\_E](#_人脸成员证件类型枚举) |
| szNumber | 证件号 |
| byRes | 保留字段 |

### 区域坐标结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwLeft | 左坐标 |
| udwTop | 上坐标 |
| udwRight | 右坐标 |
| udwButtom | 下坐标 |

### 人脸抓拍信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwFaceId | 人脸ID |
| pcPicBuff | 图片缓存 |
| udwPicBuffLen | 图片缓存长度 |
| enImgType | 图像类型，参考枚举[NETDEV\_TMS\_PERSION\_IMAGE\_TYPE\_E](#_图片类型枚举枚举) |
| enImgFormat | 图像格式，参考枚举[NETDEV\_TMS\_PERSION\_IMAGE\_FORMAT\_E](#_图片格式枚举) |
| stFacePos | 人脸坐标---画面坐标归一化：0-10000 ; 矩形左上和右下点："138,315,282,684" |
| udwImageWidth | 图像宽度 |
| udwImageHeight | 图像高度 |
| szCamerID | 相机编号 |
| szRecordID | 记录ID号 |
| szTollgateID | 卡口编号 |
| szPassTime | 经过时刻,YYYYMMDDHHMMSSMMM，时间按24小时制。第一组MM表示月，第二组MM表示分，第三组MMM表示毫秒 |
| udwFaceNum | 人脸个数 全景图时有效 |
| szIPAddr | 设备IP地址 |
| bRes | 保留字段 |

**See also：**

[NETDEV\_FaceSnapshotCallBack\_PF](#_注册人脸抓拍上报回调函数)

### 人脸位置信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwTopLeftX | 左上角X [0, 10000] Upper left corner X [0, 10000] |
| dwTopLeftY | 左上角Y [0, 10000] Upper left corner Y [0, 10000] |
| dwBottomRightX | 右下角X [0, 10000] Lower right corner x [0, 10000] |
| dwBottomRightY | 右下角Y [0, 10000] Lower right corner y [0, 10000] |
| bRes | 保留字段 |

### 热度图报文数据信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPicNum | 区域个数 |
| stTmsXmlInfo | XML信息 |
| stTmsPicInfo | 区域信息 |

**See also：**

[NETDEV\_HeatMapCallBack\_PF](#_注册热度图上报回调函数)

### 热度图报文XML解析信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szDevID | 设备编号 |
| szRecordID | 记录ID号 |
| szCollectTime | 采集时间 |
| dwColumns | 检测列数 |
| dwRows | 检测行数 |
| dwIntervalTime | 检测间隔 |
| szIPAddr | 设备IP地址 |
| bRes | 保留字段 |

### 热度图报文区域解析信息结构体

|  |
| --- |
| typedef struct tagNETDEVTMSHeatMapPicInfo  {  UINT32 udwPicSize;  CHAR \*pcPicData;  }NETDEV\_TMS\_HEAT\_MAP\_PIC\_INFO\_S,\*LPNETDEV\_TMS\_HEAT\_MAP\_PIC\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPicSize | 区域大小 |
| pcPicData | 区域数据 |

### 人员报警信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 通知记录ID |
| udwTimestamp | 通知上报时间UTC格式，单位秒 |
| udwNotificationType | 通知类型0：实时通知：历史通知 |
| udwFaceInfoNum | 人脸信息数目范围：[0, 1] |
| stCtrlFaceInfo | 人脸信息列表，当采集信息没有人脸时，可不带FaceInfo相关字段 |
| udwFinishFaceNum | 人脸结束数量范围：[0, 40] |
| audwFinishFaceList | 人脸结束列表 |
| szReference | 描述信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_PersonAlarmMessCallBack\_PF](#_注册速人脸报警消息回调函数)

### 人脸通过记录信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwRecordID | 人脸通行记录ID |
| udwType | 人脸通行类型，参见枚举[NETDEV\_FACE\_PASS\_RECORD\_TYPE\_E](#_人脸通行记录类型枚举) |
| tPassingTime | 过人时间，UTC格式，单位秒 |
| udwChannelID | 通道ID |
| szChlName | 抓拍通道名称，范围[1,63] |
| stCompareInfo | 比对信息 |
| byRes | 保留字段 |

### 人脸比对信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSimilarity | 相似度 |
| stPersonInfo | 人脸库成员信息 |
| stPanoImage | 人脸大图 |
| stFaceImage | 人脸小图 |
| stFaceArea | 人脸区域坐标 |
| udwCapSrc | 采集来源 |
| udwFeatureNum | 半结构化特征数目如果没有半结构化特征，可不带相关字段PTS必带 |
| pstFeatureInfo | 半结构化特征列表如果没有半结构化特征，可不带相关字段 PTS必带 |
| stFaceAttr | 人脸属性信息 |
| stPersonAttr | 关联人员属性信息 |
| byRes | 保留字段 |

### 半结构化特征信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szFeatureVersion | 人脸半结构化特征提取算法版本号[0, 20] |
| szFeature | 基于人脸提取出来的特征信息目前加密前B |
| byRes | 保留字节 |

### 人脸属性信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwGender | 性别详见[NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举) |
| udwAgeRange | 年龄段详见[NETDEV\_AGE\_RANGE\_E](#_年龄段枚举) |
| udwGlassFlag | 是否戴眼镜标志详见[NETDEV\_GLASS\_FLAG\_E](#_是否戴眼镜标志枚举) |
| udwGlassesStyle | 眼镜款式详见[NETDEV\_GLASSES\_STYLE\_E](#_眼镜款式枚举) |
| udwMask | 口罩详见[[NETDEV\_MASK\_FLAG\_E](#_是否戴口罩枚举)](#_是否戴口罩枚举_1) |
| fTemperature | 体温单位：摄氏度精度：小数点后位 |
| udwEmotion | 情绪情况，未检测时可选 参见 [NETDEV\_EMOTION\_FLAG\_E](#_情绪情况) |
| udwSmile | 是否微笑，未检测时可选 详见 [NETDEV\_SMILE\_FLAG\_E](#_微笑标志) |
| udwAttractive | 颜值，未检测时可选 数值范围[0~100] |
| udwSkinColor | 肤色，未检测时可选 详见 [NETDEV\_SKINCOLOR\_TYPE\_E](#_肤色枚举) |
| udwBeard | 胡子，未检测时可选 详见 [NETDEV\_BEARD\_FLAG\_E](#_胡子标志) |
| byRes | 保留字段 |

### 人员属性

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwGender | 性别详见[NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举) |
| udwAgeRange | 年龄段详见[NETDEV\_AGE\_RANGE\_E](#_年龄段枚举) |
| udwSleevesLength | 上衣长短款式详见[NETDEV\_SLEEVES\_LENGTH\_E](#_上衣长短款式枚举) |
| udwCoatColor | 上衣颜色详见[NETDEV\_CLOTHES\_COLOR\_E](#_衣服颜色枚举) |
| udwTrousersLength | 下衣长短款式，[NETDEV\_TROUSERS\_LENGTH\_E](#_下衣长短款式枚举) |
| udwTrousersColor | 下衣颜色详见[NETDEV\_CLOTHES\_COLOR\_E](#_衣服颜色枚举) |
| udwBodyToward | 身体抓怕朝向详见[NETDEV\_BODY\_TOWARD\_E](#_身体抓拍朝向枚举) |
| udwShoesTubeLength | 鞋子长短款式详见[NETDEV\_SHOES\_TUBE\_LENGTH\_E](#_鞋子长短款式枚举) |
| udwHairLength | 发型长短详见[NETDEV\_HAIR\_LENGTH\_E](#_发型长短枚举) |
| udwBagFlag | 是否携包标志详见[NETDEV\_BAG\_FLAG\_E](#_是否携包标志枚举) |
| fTemperature | 体温单位：摄氏度精度：小数点后位 |
| udwMask | 口罩详见[NETDEV\_PERSON\_MASK\_FLAG\_E](#_是否戴口罩枚举_1) |
| udwCoatTexture | 上衣纹理详见[NETDEV\_CLOTHES\_TEXTURE\_E](#_上衣纹理枚举) |
| udwMovingDirection | 人员运动方向详见[NETDEV\_MOVE\_DIRECTION\_E](#_人员运动方向枚举) |
| byRes | 保留字段 |

### 结构化告警上报信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szReference | 描述信息 |
| udwTimeStamp | 告警时间从年月日点开始的秒数 |
| udwSeq | 告警序号 |
| udwSrcID | 告警源ID |
| szSrcName | 告警源名称 |
| udwNotificationType | 通知类型0：实时通知1：历史通知 |
| szDeviceID | 告警设备ID，国标协议接入时填写国标注册码。长度[1,32] |
| szRelatedID | 关联ID，告警和数据关联；或多通道目标数据的关联，同一个相机内全局唯一。长度为15个字符 |
| byRes | 保留字段 |

### 结构化数据信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stObjectInfo | 目标信息 |
| udwImageNum | 图像个数 |
| pstImageInfo | 图像相关信息需动态申请内存 |
| byRes | 保留字段 |

### 目标信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwFaceNum | 人脸数量 |
| pstFaceInfo | 人脸信息需动态申请内存 |
| udwPersonNum | 人员数量 |
| pstPersonInfo | 人员信息需动态申请内存 |
| udwNonMotorVehNum | 非机动车数量 |
| pstNonMotorVehInfo | 非机动车信息需动态申请内存 |
| udwVehicleNum | 车辆数量 |
| pstVehInfo | 车辆信息需动态申请内存 |
| byRes | 保留字段 |

### 人脸信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwFaceID | 人脸ID |
| udwFaceDoforPersonID | 人脸所属人员ID |
| szPosition | 人脸位置信息 |
| udwSmallPicAttachIndex | 人脸对应的小图在图像列表中的索引 |
| udwLargePicAttachIndex | 人脸对应的大图在图像列表中的索引 |
| szFeaturVersion | 半结构化特征厂商类型版本号 |
| szFeature | 基于人脸提取出来的特征信息采用base64编码前加密前Bytes |
| stFaceAttr | 人脸属性信息 |
| pstRuleInfo | 规则信息 需动态申请内存 |
| udwFaceDoforNonMotorID | 人脸所属非机动车ID |
| udwFaceDoforVehicleID | 人脸所属机动车ID |
| byRes | 保留字段 |

**See also：**

[NETDEV\_OBJECT\_INFO\_S](#_目标信息)

### 人员信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPersonID | 人员ID |
| udwPersonDoforFaceID | 人员所属人脸ID |
| szPosition | 人员位置信息 |
| udwSmallPicAttachIndex | 人员对应的小图在图像列表中的索引 |
| udwLargePicAttachIndex | 人员对应的大图在图像列表中的索引 |
| szFeaturVersion | 半结构化特征厂商类型版本号 |
| szFeature | 半结构化特征信息采用base64编码加密前Bytes |
| stPersonAttr | 人员信息 |
| pstRuleInfo | 规则信息 需动态申请内存 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_OBJECT\_INFO\_S](#_目标信息)

### 非机动车信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 非机动车ID |
| szPosition | 非机动车位置信息 |
| udwSmallPicAttachIndex | 非机动车对应的小图在图像列表中的索引 |
| udwLargePicAttachIndex | 非机动对应的大图在图像列表中的索引 |
| stNoMotorVehAttr | 非机动车属性信息 |
| udwPersonOnNoVehiNum | 驾乘人员数目 |
| pstPersonAttr | 人员属性，需动态申请内存 |
| pstRuleInfo | 规则信息 需动态申请内存 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_OBJECT\_INFO\_S](#_目标信息)

### 非机动车属性信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSpeedType | 结构化场景中非机动车速度类型详见[NETDEV\_SPEED\_TYPE\_E](#_结构化场景中非机动车速度类型枚举) |
| udwImageDirection | 结构化场景中非机动车相对画面运动方向详见  [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 | 非机动车类型详见[NETDEV\_NON\_VEH\_TYPE\_E](#_非机动车类型枚举) |
| byRes | 保留字段 |

### 车辆信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 车辆ID |
| szPosition | 车辆位置信息 |
| udwSmallPicAttachIndex | 车辆对应的小图在图像列表中的索引 |
| udwLargePicAttachIndex | 车辆对应的大图在图像列表中的索引 |
| udwPlatePicAttachIndex | 车牌对应的小图在图像列表中的索引 |
| szFeatureVersion | 半结构化特征厂商类型版本号 |
| szFeature | 基于人脸提取出来的特征信息采用base64编码加密前Bytes |
| stVehAttr | 车辆属性信息 |
| stPlateAttr | 车牌属性信息 |
| pstRuleInfo | 规则信息 需动态申请内存 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_OBJECT\_INFO\_S](#_目标信息)

### 车辆属性信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 车辆类型详见[NETDEV\_VEHICLE\_TYPE\_E](#_车辆类型枚举) |
| udwColor | 车身颜色详见[NETDEV\_PLATE\_COLOR\_E](#_车牌颜色枚举) |
| udwSpeedUnit | 车辆速度单位0：公里/每小时1：英里/每小时 |
| fSpeedValue | 车辆速度 |
| udwSpeedType | 结构化场景中的机动车车辆速度类型详见[NETDEV\_SPEED\_TYPE\_E](#_结构化场景中非机动车速度类型枚举) |
| szVehicleBrand | 车辆车标编码（自行编码) |
| udwImageDirection | 结构化场景中的机动车在画面坐标系中的行驶方向详见  [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 | 保留字段 |

### 车牌属性信息

|  |
| --- |
| typedef struct tagNETDEVPlateAttr  {  CHAR szPlateNo[NETDEV\_LEN\_64];  UINT32 udwColor;  UINT32 udwType;  BYTE byRes[128];  }NETDEV\_PLATE\_ATTR\_S,\*LPNETDEV\_PLATE\_ATTR\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szPlateNo | 车牌号码 |
| udwColor | 车牌颜色详见[NETDEV\_PLATE\_COLOR\_E](#_车牌颜色枚举) |
| udwType | 车牌类型详见[NETDEV\_PLATE\_TYPE\_E](#_车牌类型枚举) |
| byRes | 保留字段 |

### 车辆识别事件结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szReference | 订阅者描述信息 |
| udwSrcID | 告警源ID |
| stVehicleEventInfo | 车辆比对报警信息 需动态申请内存 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_VehicleAlarmMessCallBack\_PF](#_注册车牌识别报警回调函数)

### 车辆比对报警信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 通知记录ID |
| udwTimestamp | 通知上报时间，UTC格式，单位秒 |
| udwNotificationType | 通知类型 详见 [NETDEV\_NOTIFICATION\_TYPE\_E](#_通知类型枚举) |
| udwVehicleInfoNum | 车辆信息数目 [0, 1] |
| pstVehicleRecordInfo | 车辆信息列表 |
| byRes | 保留字段 |

### 车辆识别记录信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwRecordID | 车辆识别记录ID |
| udwChannelID | 通道ID，抓拍推送时有效 |
| udwPassingTime | 过车时间，UTC格式，单位秒 |
| szChannelName | 卡口相机名称 |
| stPlateAttr | 车牌抓拍信息 |
| stVehAttr | 车辆抓拍信息 |
| stPlateImage | 车牌抓拍图片 图片加密后大小不超过1M |
| stVehicleImage | 车辆抓拍图片 结构化查询时携带 图片加密后大小不超过1M |
| stPanoImage | 全景图 结构化查询时携带 仅携带图片URL和size，图片数据需要通过/LAPI/V1.0/System/Picture接口获取 |
| stMonitorAlarmInfo | 车牌告警布控信息 |
| byRes | 保留字段 |

### 车牌信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szPlateNo | 车牌号 |
| udwColor | 车牌颜色 参见[NETDEV\_PLATE\_COLOR\_E](#_车牌颜色枚举) |
| udwType | 车牌类型，参见[NETDEV\_PLATE\_TYPE\_E](#_车牌类型枚举) |
| byRes | 保留字段 |

### 订阅智能事件信息结构体

|  |
| --- |
| typedef struct tagstNETDEVSubscribeSmartInfo  {  UINT32 udwNum;  UINT32 \*pudwSmartType;  BYTE byRes[128];  }NETDEV\_SUBSCRIBE\_SMART\_INFO\_S, \*LPNETDEV\_SUBSCRIBE\_SMART\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 订阅智能告警数 |
| pudwSmartType | 订阅的智能告警类型 参见枚举 [NETDEV\_SMART\_ALARM\_TYPE\_E](#_智能告警类型枚举) ，根据udwNum动态申请 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SubscribeSmart](#_订阅智能事件)

### 智能事件信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道ID |
| udwSubscribeID | 订阅ID |
| udwSubscribeType | 订阅类型 |
| udwCurrrntTime | 当前时间，UTC格式，单位秒 |
| udwEndTime | 结束时间，UTC格式，单位秒 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_SubscribeSmart](#_订阅智能事件)、[NETDEV\_UnsubscribeSmart](#_取消订阅智能事件)

### Lapi告警订阅信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 订阅类型 按位表示，详见 [NETDEV\_ALARM\_TYPE\_V30\_E](#_告警支持类型枚举) |
| udwLibIDNum | 订阅的库ID数目 LibIDNum为0xffff时 表示订阅所有库 |
| audwLibIDList | 订阅的库ID列表 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_SubscibeLapiAlarm](#_LAPI告警订阅)

### 订阅信息成功返回信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 订阅ID |
| udwCurrrntTime | 当前时间，UTC格式，从1970年1月1日0点开始的秒数 |
| udwTerminationTime | 结束时间，UTC格式，从1970年1月1日0点开始的秒数 |
| udwSupportAlarmType | 请求消息携带订阅告警类型时返回值需携带此参数，返回0说明响应未携带该数据 |
| szReference | 订阅者描述信息 以URL格式体现 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_SubscibeLapiAlarm](#_LAPI告警订阅)

### 告警记录返回信息（人脸识别和车牌识别）结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTotal | 告警记录总数 |
| udwOffset | 记录偏移量 |
| udwNum | 此次返回告警记录个数 |
| byRes | 保留字段 |

**See also：**

[[NETDEV\_FindFaceRecordDetailList](#_查询人脸识别记录)](#_查询人脸识别记录)

### 停车场出入口信息列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 出入口数量 |
| astParkEntranceInfo | 出入口信息 一个停车场最大支持16个出入口 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_GetParkEntranceInfoList](#_获取某停车场下所有出入口信息)

### 停车场出入口信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwParkEntranceID | 出入口ID 添加出入口是不携带 |
| szName | 出入口名称，长度范围[1,64] |
| udwVehicleLaneNum | 车道数，最多2个 |
| byRes | 保留字段 |

### 停车场信息列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 停车场数量 |
| astParkInfo | 停车场信息 一个VMS最大支持8个停车场 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_GetParkInfoList](#_批量获取停车场信息)

### 停车场信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwParkingLotID | 停车场ID |
| szParkingLotName | 停车场名称，长度范围[1,64] |
| udwTotalSpaceNum | 总车位数 |
| udwRmnSpaceNum | 剩余车位数 |
| udwReserveSpaceNum | 固定车位数 |
| udwRmnReserveSpaceNum | 剩余固定车位数 |
| udwMainParkingLotID | 所属停车场ID 没有所属主库时可不携带该字段 |
| byRes | 保留字段 |

### 停车场出入口ID结构体

|  |
| --- |
| typedef struct tagstNETDEVParkEntranceID  {  UINT32 udwParkID;  UINT32 udwEntranceID;  BYTE byRes[256];  }NETDEV\_PARK\_ENTRANCE\_ID\_S,\*LPNETDEV\_PARK\_ENTRANCE\_ID\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwParkID | 停车场ID |
| udwEntranceID | 出入口ID |
| byRes | 保留字段 |

**See also：**

[NETDEV\_GetParkLaneInfoList](#_获取停车场出入口所有车道信息)

### 车道信息列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 车道数，最多2车道 |
| astVehLaneInfo | 车道信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_GetParkLaneInfoList](#_获取停车场出入口所有车道信息)

### 车道信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwVehLaneID | 车道ID |
| szName | 车道名称 |
| udwType | 车道类型 详见 [NETDEV\_VEH\_LANE\_TYPE\_E](#_车道类型枚举) |
| udwNum | 自定义启用时间段个数，最大4段 |
| astEnTmSectInfo | 自定义启用时间段列表 |
| stTmpVehThrough | 临时车放行 |
| stRegVehThrough | 固定车放行 |
| udwDevNum | 设备数量 最多4个 |
| audwDevID | 车道关联的出入口设备ID列表 |
| byRes | 保留字段 |

### 自定义时间段结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szBegin | 开始时间, 格式hh:mm:ss 长度范围[0,31] |
| szEnd | 结束时间，格式hh:mm:ss 长度范围[0,31] |
| byRes | 保留字段 |

### 车辆放行信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 放行方式 详见[NETDEV\_VEH\_RELEASE\_TYPE\_E](#_车辆放行类型枚举) |
| udwNum | 自定义放行时间段列表 最多4个 |
| astTmSectInfo | 自定义放行时间段信息 |
| byRes | 保留字段 |

### 停车场支付订单信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szOrderNo | 订单号，由第三方生成 |
| udwOrderStatus | 订单支付状态 详见枚举 [NETDEV\_ORDER\_STATUS\_E](#_订单支付状态枚举) |
| szPlateNo | 车牌号 |
| udwPayType | 支付类型 详见枚举 [NETDEV\_PAY\_TYPE\_E](#_支付类型枚举) |
| szPayTradeNo | 支付订单号 |
| udwTotalAmount | 停车费总金额，单位分 |
| udwDiscountAmount | 优惠减免的金额，单位分 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_IssuePayOrder](#_下发第三方支付订单)

### 停车花费结构体

|  |
| --- |
| typedef struct tagstNETDEVParkCost  {  UINT32 udwTotalAmount;  UINT32 udwEnterTime;  UINT32 udwExitLimitTime;  BYTE byRes[256];  }NETDEV\_PARK\_COST\_S, \*LPNETDEV\_PARK\_COST\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTotalAmount | 优惠减免的金额，单位分 |
| udwEnterTime | 进场时间，UTC时间，单位秒 |
| udwExitLimitTime | 规定离场时间，单位分钟 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_IssuePayOrder](#_下发第三方支付订单)

### 停车场事件信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 记录ID |
| udwParkID | 停车场ID |
| szPlateNo | 车牌号码 |
| stVehAttr | 车辆属性信息 |
| udwVehRemainNum | 场内车辆数 |
| udwIsWhiteList | 是否是白名单车辆 0：否 1：是 |
| udwIsBlackList | 是否是黑名单车辆 0：否 1：是 |
| szEntranceName | 岗亭出入口名称 |
| szLaneName | 车道名称 |
| udwCrossDirection | 过车方向 0：入场 1：出场 |
| pstEnterInfo | 入场信息 |
| pstExitInfo | 出场信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_ParkEventMessCallBack\_PF](#_注册停车场事件回调函数)

### 入场车辆抓拍信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwEnterTime | 入场时间，UTC时间，单位秒 |
| stPlateImage | 车牌图片 |
| stVehImage | 车辆图片 |

### 出场车辆抓拍信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwExitTime | 出场时间(UTC时间，单位秒) |
| udwTotalAmount | 停车费(单位分) |
| udwIsAlreadyPaid | 是否已缴费 0:否 1:是 |
| udwChargeRecordID | 缴费记录ID |
| stPlateImage | 车牌图片 |
| stVehImage | 车辆图片 |
| szOperatorName | 操作员 |
| byRes | 保留字段 |

### 时间模板配置结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTemplateID | 时间模板ID |
| szTemplateName | 时间模板名称 [1, 63] |
| szTemplateDesc | 时间模板描述 [1, 128] |
| udwLastChange | 时间模板最后修改时间 |
| stWeekPlanInfo | 布控任务布防计划 |
| stExceptionInfo | 布控任务例外计划 |
| bIsBuiltin | 是否为内置时间模板 仅VMS支持 1:是 0:否 |
| udwTemplateType | 时间模板类型 仅VMS支持 0:录像时间模板 1:报警时间模板 2:用户时间模板 |
| byRes | 保留字节 |

**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)

### 布控任务例外计划结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 例外日期是否使能 0:不使能 1：使能 |
| udwNum | 例外日期个数 [0, 16] |
| stExceptionDayInfo | 每天的布防计划具体信息 |
| byRes | 保留字节 |

### 每天的布防计划具体信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 例外日期索引 |
| bEnabled | 例外日期是否使能 |
| szDate | 例外日期 |
| udwNum | 例外时间段个数 NVR最大为8段 IPC/PTS最大为4段 |
| stTimeSectionInfo | 布防配置具体信息 |
| byRes | 保留字节 |

### 通用ID列表结构体

|  |
| --- |
| typedef struct tagNETDEVIDList  {  UINT32 udwNum;  UINT32 \*pudwIDs;  BYTE byRes[256];  }NETDEV\_ID\_LIST\_S, \*LPNETDEV\_ID\_LIST\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 数量 |
| pudwIDs | ID列表 Malloc申请内存 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_BatchDeleteTimeTemplate](#_批量删除指定的时间模板)

### 时间模板结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTemplateID | 模板ID |
| szTemplateName | 模板名称 |
| udwLastChange | 最后的修改时间 |
| byRes | 保留字段 |

**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)

### 门禁人员信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwReqSeq | 请求序号 |
| udwPersonID | 人员编号 |
| szName | 姓名 |
| udwGender | 性别 参见枚举 [NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举) |
| stMemberIDInfo | 证件信息 |
| szTelephone | 联系电话 |
| szAddress | 地址 |
| szDesc | 备注 |
| udwCardNum | 门禁卡个数，取值范围[1,6] |
| stACSPersonCardList | 门禁卡信息 |
| stFaceImage | 人脸图片 |
| udwType | 人员类型 参见[NETDEV\_ACS\_PERSON\_TYPE\_E](#_人员类型枚举) |
| stStaffInfo | 员工信息 |
| stVisitor | 访客信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_ACSPersonCtrl](#_门禁人员管理)

### 图片信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 照片数 |
| stImageList | 人脸照片列表 |
| udwMajorImageIndex | 主照片索引 |
| byRes | 保留字段 |

### 员工信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szNumber | 人员编号 |
| szBirthday | 出生日期 |
| szDeptName | 部门名称 |
| udwDeptID | 部门ID |

### 时间信息结构体

|  |
| --- |
| typedef struct tagACSTimeSection  {  INT64 tStartTime;  INT64 tEndTime;  BYTE byRes[32];  }NETDEV\_ACS\_TIME\_SECTION\_S, \*LPNETDEV\_ACS\_TIME\_SECTION\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| tStartTime | 起始时间 UTC时间 单位秒s |
| tEndTime | 结束时间 UTC时间 单位秒s |
| byRes | 保留字段 |

### 访客信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szVisitorCompany | 访客公司 [1,64]字符 |
| udwVisitorCount | 访客人数 |
| udwIntervieweeID | 被访者ID |
| szIntervieweeName | 被访者姓名 [1,64]字符 |
| szIntervieweeDept | 被访者部门 [1,64]字符 |
| tScheduleTime | 预约访问时间 |
| tRealTime | 实际到访时间 |
| udwStatus | 状态 参见枚举[NETDEV\_ACS\_VISIT\_STATUS\_E](#_访客状态枚举) |
| byRes | 保留字段 |

### 人员所持门禁卡信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwCardID | 绑定ID |
| udwCardType | 卡片类型 |
| udwCardStatus | 卡片状态 0:空白 1:激活 2:冻结 3:注销 |
| szCardNo | 卡号 |
| udwReqSeq | 序号 |
| stValidTime | 有效时间 |
| byRes | 保留字段 |

**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)

### 黑名单信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwBlackListID | 黑名单ID |
| stIdentificationInfo | 身份信息 |
| byRes | 保留字段 |

**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)

### 人员列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 人员数量 |
| pstPersonInfoList | 员工信息列表 根据udwNum动态申请 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_AddACSPersonList](#_批量添加人员信息)

### 授权信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPermissionID | 权限ID |
| szPermissionName | 权限名称 |
| udwPermissionType | 权限类型：0表示员工权限组，1表示访客权限组 |
| stPersonList | 人员ID列表，其中dwSize为人员个数 |
| udwTemplateID | 时间模板ID |
| stValidTime | 有效时间 |
| stDoorList | 门通道列表, 其中dwSize为门通道个数 |

**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)

### 人脸识别模块批量操作列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 批量操作数量 |
| pstBatchList | 批量操作信息 根据udwNum进行动态申请 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_DeleteACSPersonList](#_批量删除门禁人员信息)

### 人脸识别模块批量操作信息结构体

|  |
| --- |
| typedef struct tagNETDEVFaceBatchInfo  {  UINT32 udwReqSeq;  UINT32 udwResultCode;  UINT32 udwID;  BYTE byRes[128];  }NETDEV\_FACE\_BATCH\_INFO\_S,\*LPNETDEV\_FACE\_BATCH\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwReqSeq | 请求数据序号 |
| udwResultCode | 返回错误码 |
| udwID | 编号 |
| byRes | 保留字段 |

### 出入记录信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwAlarmType | 告警类型 |
| tTimeStamp | 告警时间 |
| szDoorName | 门名称 |
| szDoorNo | 门编号 |
| udwDoorDirect | 进出方向 0:进,1:出 |
| szCardNo | 刷卡卡号 |
| szPersonName | 刷卡人姓名 |
| udwPersonType | 人员类型，参见[NETDEV\_ACS\_PERSON\_TYPE\_E](#_人员类型枚举) |
| szPersonPhone | 刷卡人电话 |
| szPersonDept | 刷卡人部门 |
| stCompareInfo | 脸对比信息，速通门会携带此信息 |
| stFaceAttr | 人脸属性信息 |
| udwCardStatus | 卡状态 0：未注册 1：已注册 |
| stLibMatchInfo | 库对比信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_FindNextACSAttendanceLog](#_获取下一条出入记录)

### 人脸对比信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stPersonImage | 人员图片 |
| stSnapshotImage | 抓拍图片 |
| byRes | 保留字段 |

### 查询条件结构体

|  |
| --- |
| typedef struct tagNETDEVPagedQueryInfo  {  UINT32 udwLimit;  UINT32 udwOffset;  BYTE byRes[128];  }NETDEV\_PAGED\_QUERY\_INFO\_S,\*LPNETDEV\_PAGED\_QUERY\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwLimit | 每次查询的数量 |
| udwOffset | 从当前序号开始查询 |
| byRes | 保留字节 |

**See also：**

[[NETDEV\_FindACSPersonBlackList](#_获取访客黑名单列表)](#_获取访客黑名单列表)

### 门禁人员基本信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPersonID | 人员编号 |
| szName | 姓名 |
| udwGender | 性别 参见枚举 [NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举) |
| stMemberIDInfo | 证件信息 |
| szTelephone | 联系电话 |
| udwCardID | 卡片编号 |
| szCardNo | 卡片号码 |
| udwType | 人员类型 0员工 1访客 |
| stStaffInfo | 员工信息 |
| stVisitor | 访客信息 |
| byRes | 保留字段 |

**See also：**

[NETDEV\_FindNextACSPersonInfo](#_获取下一个门禁人员信息)

### 访客记录信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwLogID | 日子ID |
| udwVisitorID | 访客ID |
| szVisitorName | 访客姓名 [1,64]字符 |
| szVisitorCompany | 访客公司 [1,64]字符 |
| szVisitorPhone | 访客电话 |
| szCardNo | 访客卡号 |
| udwIntervieweeID | 被访者ID |
| szIntervieweeName | 被访者姓名 [1,64]字符 |
| szIntervieweeDept | 被访者部门 [1,64]字符 |
| tScheduleStartTime | 预约来访时间 UTC时间 单位秒s |
| tRealStartTime | 实际来访时间 UTC时间 单位秒s |
| udwStatus | 状态 参见枚举[NETDEV\_ACS\_VISIT\_STATUS\_E](#_访客状态枚举) |
| byRes | 保留字段 |

**See also：**

[[NETDEV\_FindNextACSVisitLog](#_获取下一条访客记录)](#_获取下一条访客记录)

### 门授权信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPermissionNum | 权限组个数 |
| stValidTime | 有效时间 |
| pstPermissionGroupList | 权限组信息列表.Num数为0时可选 |
| byRes | 保留字段 |

**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)

### 权限组信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwPermissionGroupID | 权限组ID |
| szPermissionGroupName | 权限组名称 |
| byRes | 保留字段 |

### 人员核验结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szReference | 用于客户端确认推送消息的url |
| udwSeq | 通知记录序号 |
| udwChannelID | 通道ID VMS支持 |
| szChannelName | 通道名称，长度 [1,64]，VMS支持 |
| udwTimestamp | 通知上报时间 UTC格式，单位秒 |
| udwNotificationType | 通知类型 0：实时通知1：历史通知 |
| udwFaceInfoNum | 人脸信息数目 范围：[0, 1] |
| pstCtrlFaceInfo | 人脸信息 需动态申请内存 |
| udwCardInfoNum | 卡信息数目 范围：[0, 1] |
| pstCtrlCardInfo | 卡信息 需动态申请内存 |
| udwGateInfoNum | 闸机信息数目 范围：[0, 1] |
| pstCtrlGateInfo | 闸机信息 需动态申请内存 |
| udwLibMatInfoNum | 库比对信息数目 范围：[0, 16] |
| pstLibMatchInfo | 库比对信息 需动态申请内存 |
| udwTemperatureInfoNum | 温度信息数目 |
| pstTemperatureInfo | 温度信息列表，需根据温度信息数目动态申请内存 |
| byRes | 保留字节 |

**See also：**

[NETDEV\_AlarmMessFGCallBack\_PF](#_注册速通门报警消息回调函数)

### 人脸信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 记录ID |
| udwTimestamp | 采集时间 UTC格式，单位秒 |
| udwCapSrc | 采集来源 详见 [NETDEV\_CAP\_SRC\_E](#_采集来源枚举) FaceInfo选择1 |
| udwFeatureNum | 半结构化特征数目 范围：[0, 2] |
| pstFeatureInfo | 半结构化特征列表 需动态分配内存 |
| stPanoImage | 人脸全景图 |
| stFaceImage | 人脸小图 |
| stFaceArea | 人脸全景图人脸区域坐标 |
| fTemperature | 人员体温 单位：摄氏度，注：小数点后1位 |
| udwMaskFlag | 是否戴口罩，0：未知或未启用检测；1：未戴口罩；2：戴口罩 |
| byRes | 保留字节 |

### 卡信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 记录ID |
| udwTimestamp | 采集时间 UTC格式，单位秒 |
| udwCapSrc | 采集来源 详见 [NETDEV\_CAP\_SRC\_E](#_采集来源枚举) CardInfo选择2或3 |
| udwCardType | 0：身份证，1：门禁卡 |
| szCardID | 门禁卡字段：物理卡号 最长18位 |
| udwCardStatus | 门禁卡字段：卡状态 |
| szName | 身份证字段：姓名 范围[1,63] |
| udwGender | 身份证字段：性别 详情参见枚举[NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举) |
| szBirthday | 身份证字段：出生日期 YYYYMMDD |
| szResidentialAddress | 身份证字段：住址 |
| szIdentityNo | 身份证字段：身份证号码 |
| szIssuingAuthority | 身份证字段：发证机关 |
| szIssuingDate | 身份证字段：发证日期 YYYYMMDD |
| szValidDateStart | 身份证字段：证件有效期开始时间 |
| szValidDateEnd | 身份证字段：证件有效期结束时间 |
| stIDImage | 身份证照片 |
| byRes | 保留字节 |

### 闸机信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 记录ID |
| udwTimestamp | 采集时间 |
| udwCapSrc | 采集来源 详见 [NETDEV\_CAP\_SRC\_E](#_采集来源枚举) GateInfo选择4 |
| udwInPersonCnt | 进入人员计数 |
| udwOutPersonCnt | 离开人员计数 |
| byRes | 保留字节 |

### 库比对信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 记录ID |
| udwLibID | 库ID |
| udwLibType | 库类型 |
| udwMatchStatus | 匹配状态 详见[NETDEV\_MATCH\_STATUS\_E](#_匹配状态枚举) |
| udwMatchPersonID | 匹配人员ID |
| udwMatchFaceID | 匹配人脸ID |
| stMatchPersonInfo | 匹配人员信息 |
| byRes | 保留字节 |

### 匹配人员信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szPersonName | 成员名字 范围[1,63] |
| udwGender | 成员性别 详情参见枚举[NETDEV\_GENDER\_TYPE\_E](#_成员性别枚举) |
| szCardID | 门禁卡号 |
| szIdentityNo | 身份证卡号 |
| szPersonCode | 人员编码 可填写学号或工号 范围:[1, 15] |
| byRes | 保留字节 |

### 图像相关信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwIndex | 图像索引 |
| udwType | 图像类型 |
| udwFormat | 图像格式 详见 [NETDEV\_IMAGE\_FORMAT\_E](#_图像格式枚举) |
| udwWidth | 图像的宽度 |
| udwHeight | 图像的高度 |
| udwCaptureTime | 图片采集时刻 |
| szUrl | 图片URL |
| udwSize | 图像经过base64编码之后的长度 最大3M |
| pszData | 图像的base64编码数据 |
| byRes | 保留字段 |

### 车牌告警布控信息结构体

|  |
| --- |
| typedef struct tagstNETDEVMonitorAlarmInfo  {  UINT32 udwMonitorReason;  UINT32 udwMonitorAlarmType;  UINT32 udwMemberID;  }NETDEV\_MONITOR\_ALARM\_INFO\_S,\*LPNETDEV\_MONITOR\_ALARM\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwMonitorReason | 布控原因类型 |
| udwMonitorAlarmType | 布控告警类型 0：匹配告警 1：不匹配告警 |
| udwMemberID | 车辆成员ID |

### 车辆成员信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwReqSeq | 请求数据序列号 |
| udwMemberID | 人脸成员ID |
| stPlateAttr | 车牌信息 |
| stVehicleAttr | 车辆信息 |
| bIsMonitored | 是否已布控 0未布控 1已布控 |
| udwDBNum | 所属车辆库数量 |
| audwDBIDList | 所属车辆库ID数组 |
| byRes | 保留字段 |

### 车辆信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwColor | 车身颜色 详见枚举[NETDEV\_PLATE\_COLOR\_E](#_车牌颜色枚举) |
| stVehicleImage | 车辆图片 图片加密后大小不超过4M |

### 车牌信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szPlateNo | 车牌号 |
| udwColor | 车牌颜色 参见[NETDEV\_PLATE\_COLOR\_E](#_车牌颜色枚举) |
| udwType | 车牌类型，参见[NETDEV\_PLATE\_TYPE\_E](#_车牌类型) |
| byRes | 保留字段 |

### 车辆信息列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwVehicleNum | 车辆成员数量 |
| pstMemberInfoList | 车辆成员列表 根据udwNum进行动态申请 |
| byRes | 保留字段 |

### 批量操作成员列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTaskNo | 操作任务号，仅NVR支持 |
| udwMemberNum | 成员数量 |
| pstMemberIDList | 成员列表 根据udwNum进行动态申请 |
| byRes | 保留字段 |

### 告警记录返回信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTotal | 告警记录总数 |
| udwOffset | 记录偏移量 |
| udwNum | 此次返回告警记录个数 |
| byRes | 保留字段 |

### 车辆属性信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 车辆类型 详见 NETDEV\_VEHICLE\_TYPE\_E |
| udwColor | 车身颜色 详见 NETDEV\_PLATE\_COLOR\_E |
| udwSpeedUnit | 车辆速度单位 0：公里/每小时 1：英里/每小时 |
| fSpeedValue | 车辆速度 |
| udwSpeedType | 结构化场景中的机动车车辆速度类型 详见 NETDEV\_SPEED\_TYPE\_E |
| szVehicleBrand | 车辆车标编码（自行编码) |
| udwImageDirection | 结构化场景中的机动车在画面坐标系中的行驶方向 详见 NETDEV\_IMAGE\_DIRECTION\_E |
| byRes | 保留字段 |

### 车牌告警布控信息结构体

|  |
| --- |
| typedef struct tagstNETDEVMonitorAlarmInfo  {  UINT32 udwMonitorReason;  UINT32 udwMonitorAlarmType;  UINT32 udwMemberID;  }NETDEV\_MONITOR\_ALARM\_INFO\_S,\*LPNETDEV\_MONITOR\_ALARM\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwMonitorReason | 布控原因类型 |
| udwMonitorAlarmType | 布控告警类型 0：匹配告警 1：不匹配告警 |
| udwMemberID | 车辆成员ID |

**See also：**

### 建立透明通道参数结构体

|  |
| --- |
| typedef struct tagNETDEVSerialStart  {  INT32 dwSerialPort;  INT32 dwSerialNum;  BYTE byRes[512];  }NETDEV\_SERIAL\_START\_S, \*LPNETDEV\_SERIAL\_START\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSerialPort | 串口类型：1- 232串口，2- 485串口#当前仅支持485 |
| dwSerialNum | 串口编号 |
| byRes | 保留字段 |

### 用户信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwLevel | 用户等级 参见枚举 NETDEV\_USER\_LEVEL\_E |
| szUserName | 用户名 不支持修改，获取单个信息时作为入参 范围1~18 |
| szPassword | 密码 添加使用 范围0~256 获取时不返回该字段 |
| udwBasePermission | 基本权限，参见枚举[NETDEV\_USER\_BASE\_PERMISSION\_E](#_基本权限枚举) 按BIT位进行权限描述，相应BIT为1，表示该权限生效，为0表示该权限不生效 |
| udwNum | 视频输入通道数量 |
| astChnPermission | 通道权限列表 |
| byRes | 保留字段 |

### 通道权限信息结构体

|  |
| --- |
| typedef struct tagNETDEVChnPermissionInfo  {  INT32 dwChannelID;  INT32 dwPermission;  BYTE byRes[256];  }NETDEV\_CHN\_PERMISSION\_INFO\_S, \*LPNETDEV\_CHN\_PERMISSION\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwChannelID | 通道号 |
| dwPermission | 通道权限, 请参考#[NETDEV\_CHN\_PERMISSION\_TYPE\_E](#_通道权限枚举) |
| byRes | 保留字段 |

### 修改用户详细信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stUserInfo | 用户信息 |
| bIsModifyPassword | 是否修改密码, 修改时赋值 |
| bIsModifyOther | 是否管理员修改其他用户, 修改时赋值 |
| szNewPassword | 新密码(修改密码时使用)，范围0~256 不涉及密码的修改 那么该字段可不下发，如果配置时，密码为空，则下发空字段 |
| szCurrentPassword | 当前使用的密码 修改密码时携带 范围0~256 管理员修改其他用户密码只需要新密码则可，如果配置时，密码为空，则下发空字段 |
| byRes | 保留字段 |

### 用户密码信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szUserName | 用户名 |
| szNewPIN | 用户新密码 |
| szCurrentPIN | 用户原密码 |
| byRes | 保留字段 |

### 用户信息列表结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 用户数量 |
| astUserInfo | 用户列表 |
| byRes | 保留字段 |

### 组织信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwOrgID | 组织ID |
| dwParentID | 组织父节点ID |
| dwType | 类型 参见 [NETDEV\_ORG\_TYPE\_E](#_组织类型枚举) |
| szNodeName | 节点名称 |
| szDesc | 备注 |
| udwTime | 创建时间，UTC时间 |
| szUserName | 创建人 [1,64] |
| byRes | 保留字段 |

### 待删除组织信息结构体

|  |
| --- |
| typedef struct tagstNETDEVDelOrgDetailInfo  {  INT32 dwOrgNum;  INT32 \*pdwOrgIDs;  INT32 dwOrgType;  BYTE byRes[64];  }NETDEV\_DEL\_ORG\_INFO\_S, \*LPNETDEV\_DEL\_ORG\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwOrgNum | 组织数量 |
| pdwOrgIDs | 需要删除的组织ID，根据dwOrgNum 动态申请 |
| dwOrgType | 组织类型 见 [NETDEV\_ORG\_TYPE\_E](#_组织类型枚举) |
| byRes | 保留字段 |

### 删除组织响应信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwStatus | 响应状态，类型 参见 [NETDEV\_ORG\_RESPONSE\_STAUTE\_E](#_响应状态类型枚举) |
| dwNum | 响应数量 |
| pstResultInfo | 批量删除返回信息，根据删除数量动态申请 |
| byRes | 保留字段 |

### 组织下通道信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwOrgID | 组织ID |
| dwChannelsNum | 通道个数 |
| pdwChnIDs | 通道ID 指针 根据通道个数动态申请内存 |
| byRes | 保留字段 |

### 组织通道信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwOrgID | 组织ID |
| dwChannelID | 通道ID |
| szChnName | 通道名称 |
| dwChnIndex | 通道索引 |
| dwChnType | 通道类型 |
| dwChnStatus | 通道状态 |
| dwDevID | 设备ID |
| szDevName | 设备名称 |
| byRes | 保留字段 |

### 网络端口号状态信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwSize | 协议个数 |
| astUpnpPort | 协议信息 |

### 协议信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| eType | 协议类型参见枚举# [NETDEV\_PROTOCOL\_TYPE\_E](#_协议类型枚举) |
| bEnbale | 是否支持 |
| dwPort | 端口号 |
| byRes | 保留字段 |

### 布控信息查询条件结构体

|  |
| --- |
| typedef struct tagNETDEVMonitorQueryInfo  {  UINT32 udwLimit;  UINT32 udwOffset;  BOOL bIsQueryAll;  BYTE byRes[128];  }NETDEV\_MONITOR\_QUERY\_INFO\_S, \*LPNETDEV\_MONITOR\_QUERY\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwLimit | 每次查询的数量，最大20 |
| udwOffset | 从当前序号开始查询，序号从0开始 |
| bIsQueryAll | 是否查询所有，是:TRUE,否:FALSE |
| byRes | 保留字段 |

### 查找组织信息列表条件结构体

|  |
| --- |
| typedef struct tagstNETDEVOrgFindCond  {  UINT32 udwOrgType;  UINT32 udwRootOrgID;  UINT32 udwFindType;  BYTE byRes[128];  }NETDEV\_ORG\_FIND\_COND\_S, \*LPNETDEV\_ORG\_FIND\_COND\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwOrgType | 组织类型 参见[NETDEV\_ORG\_TYPE\_E](#_组织类型枚举) |
| udwRootOrgID | 根节点组织ID |
| udwFindType | 查找模式，参见[NETDEV\_ORG\_FIND\_MODE\_E](#_组织查找模式枚举) |
| byRes | 保留字段 |

### 区域人数统计规则信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 是否使能：FALSE:不使能 TRUE:使能 |
| udwReportInterval | 上报间隔时间单位：（秒） 取值范围：[1, 60] |
| udwAreaNum | 检测区域个数 |
| pstDetectAreaRuleInfo | 检测区域规则信息，需要根据udwAreaNum动态申请内存，参见 [NETDEV\_DETECT\_AREA\_RULE\_INFO\_S](#_检测区域规则信息) |
| byRes | 保留字段 |

### 检测区域规则信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwDetectAreaID | 检测区域ID号，从0开始 |
| bEnabled | 是否使能：FALSE:不使能 TRUE:使能 |
| udwAreaNum | 检测区域个数 |
| stAreaLocationInfo | 区域位置信息，参见 [NETDEV\_AREA\_LOCATION\_INFO\_S](#_区域位置信息) |
| byRes | 保留字段 |

### 区域位置信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwAreaPointNum | 绘制规则区域中点的个数，范围[3, 6] |
| pstAreaPointCoordinate | 绘制规则区域中每个顶点的坐标，坐标原点在左上角，参见 [NETDEV\_AREA\_POINT\_COORDINATE\_S](#_绘制规则区域中每个顶点的坐标) |
| byRes | 保留字段 |

### 绘制规则区域中每个顶点的坐标

|  |
| --- |
| typedef struct tagNETDEVAreaPointCoordinate  {  UINT32 udwX;  UINT32 udwY;  }NETDEV\_AREA\_POINT\_COORDINATE\_S,\*LPNETDEV\_AREA\_POINT\_COORDINATE\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwX | X轴坐标，万分比表示，范围[0,9999] |
| udwY | Y轴坐标，万分比表示，范围[0,9999] |

### 绊线人数统计规则信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 是否使能：FALSE:不使能 TRUE:使能 |
| udwReportInterval | 上报间隔时间单位：（秒） 取值范围：[1, 60] |
| udwLineNum | 绊线数量 |
| pstDetectLineRuleInfo | 检测绊线规则信息，需要根据udwLineNum动态申请内存，参见 [NETDEV\_DETECT\_LINE\_RULE\_INFO\_S](#_检测绊线规则信息) |
| byRes | 保留字段 |

### 检测绊线规则信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwDetectLineID | 检测区域ID号，从0开始 |
| bEnabled | 是否使能：FALSE:不使能 TRUE:使能 |
| udwInDirection | 进入方向 1：B->A,方位B到方位A；2:A->B,方位A到方位B |
| udwStatisticType | 统计类型，详见 [NETDEV\_STATISTIC\_TYPE\_E](#_人数统计类型) |
| stStartPointCoordinate | 线型检测区域起始点坐标，参见 [NETDEV\_AREA\_POINT\_COORDINATE\_S](#_绘制规则区域中每个顶点的坐标) |
| stEndPointCoordinate | 线型检测区域终止点坐标，参见 [NETDEV\_AREA\_POINT\_COORDINATE\_S](#_绘制规则区域中每个顶点的坐标) |
| byRes | 保留字段 |

### 绊线人数统计清零信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bTimingResetEnabled | 计数定时清零使能：FALSE:不使能 TRUE:使能 |
| szResetTime | 定时清零时间，为一天中某个时刻清零,字符串长度[0, 24]，格式如"00:00:00"。当计数定时清零使能的时候必选 |
| byRes | 保留字段 |

### 联动球机信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwLinkDomeNum | 球机个数 |
| stLinkDomeInfoList | 球机信息列表，参见 [NETDEV\_LINK\_DOME\_INFO\_S](#_联动球机信息) |
| byRes | 保留字段 |

### 联动球机信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwDomeID | Dome ID，从0开始 |
| szIPAddr | IP地址 |
| szUserName | 登录用户名，长度[0,64] |
| szPIN | 登录密码，长度[0,64] |
| udwHttpPort | HTTP端口 |
| udwZoomCoefficient | 放大系数，范围：[1-10] |
| byRes | 保留字段 |

### 系统时间配置

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwTimeZone | 时区参见枚举# [NETDEV\_TIME\_ZONE\_E](#_时区) |
| tUtcTime | UTC时间 |
| udwDateFormat | 日期格式 0：YYYY-MM-DD 年月日1：MM-DD-YYYY 月日年2：DD-MM-YYYY 日月年 |
| udwHourFormat | 时间格式0 ：小时制 1:24 小时制 |
| byRes | 保留字段 |

### 目标列表结构体

|  |
| --- |
| typedef struct tagNETDEVObjectList  {  UINT32 udwObjectType;  UINT32 udwObjectID;  BYTE byRes[256];  }NETDEV\_OBJECT\_LIST\_S, \*LPNETDEV\_OBJECT\_LIST\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwObjectType | 目标类型 参见枚举 [NETDEV\_OBJECT\_TYPE\_E](#_目标类型枚举) |
| udwObjectID | 目标ID |
| byRes | 保留字段 |

### 视图库信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwCoordinateType | 坐标类型，0:万分比坐标(范围为[0,9999])1:像素坐标 2:归一化坐标(范围0~1) NETDEV\_INVALID\_PARAM表示无效值 |
| szViidServerID | 服务器编号，字符串长度范围[1,32]，IPC支持 |
| udwConnectMode | 连接模式：0：标准模式；1：短连接模式；NETDEV\_INVALID\_PARAM表示无效值 |
| udwObjectType | 目标类型（仅视图库协议对接，且需要上报指定目标类型时可选）。按Bit位进行组合，每个Bit位对应一种目标类型，值为0代表不上报，1代表上报。Bit0：人脸，Bit1：人体，Bit2：机动车，Bit3：非机动车 |
| byRes | 保留字段 |

### 温度信息结构体

|  |
| --- |
| typedef struct tagNETDEVCtrlTemperatureInfo  {  UINT32 udwRelativeFaceID;  FLOAT fEnvTemperature;  FLOAT fTemperatureThreshold;  FLOAT fBodyTemperature;  BYTE byRes[256];  }NETDEV\_CTRL\_TEMPERATURE\_INFO, \*LPNETDEV\_CTRL\_TEMPERATURE\_INFO; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwRelativeFaceID | 关联人脸ID，若无关联人脸，填写0xffffffff |
| fEnvTemperature | 环境温度，单位：摄氏度 |
| fTemperatureThreshold | 温度阈值，单位：摄氏度 |
| fBodyTemperature | 测量体温温度，单位：摄氏度 |
| byRes | 保留字段 |

### 录像时间信息

|  |
| --- |
| typedef struct tagNETDEVRecordTime  {  UINT32 udwChlID;  INT64 tEarliestTime;  INT64 tLatestTime;  BYTE byRes[512];  }NETDEV\_RECORD\_TIME\_S, \*LPNETDEV\_RECORD\_TIME\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwChlID | 通道号 |
| tEarliestTime | 最早时间 |
| tLatestTime | 最晚时间 |
| byRes | 保留字段 |

### 录像时间列表

|  |
| --- |
| typedef struct tagNETDEVRecordTimeList  {  UINT32 udwNum;  LPNETDEV\_RECORD\_TIME\_S pstRecordTimes;  BYTE byRes[512];  }NETDEV\_RECORD\_TIME\_LIST\_S, \*LPNETDEV\_RECORD\_TIME\_LIST\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 录像时间数量 |
| pstRecordTimes | 录像时间列表，需动态分配内存，详见 [NETDEV\_RECORD\_TIME\_S](#_录像时间信息) |
| byRes | 保留字段 |

### 检测区域图形定点坐标信息

|  |
| --- |
| typedef struct tagNETDEVPointInfo  {  UINT32 udwX;  UINT32 udwY;  BYTE byRes[32];  }NETDEV\_POINT\_INFO\_S,\*LPNETDEV\_POINT\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwX | X轴坐标，范围[0,10000] |
| udwY | Y轴坐标，范围[0,10000] |
| byRes | 保留字段 |

### 规则信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwRuleType | 规则类型 参见 [NETDEV\_RULE\_TYPE\_E](#_规则类型枚举) |
| udwTrigerType | 规则触发类型 参见 [NETDEV\_RULE\_TRIGGER\_TYPE\_E](#_规则触发类型枚举) |
| udwPointNum | 规则坐标点数 |
| pstPointInfo | 检测区域图形各顶点坐标,需动态分配内存，在顶点个数为0时，可以没有 |
| byRes | 保留字段 |

### 单个目标信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwObjectType | 目标类型，参见枚举 [NETDEV\_OBJECT\_TYPE\_E](#_目标类型枚举) |
| stFaceInfo | 人脸信息 |
| stPersonInfo | 人员信息 |
| stNonMotorVehInfo | 非机动车信息 |
| stVehInfo | 车辆信 |
| byRes | 保留字段 |

### 音频输出参数

|  |
| --- |
| typedef struct tagNETDEVAudioOutputCfg  {  UINT32 udwType;  UINT32 udwGain;  BYTE byRes[256];  }NETDEV\_AUDIO\_OUTPUT\_CFG\_S,\*LPNETDEV\_AUDIO\_OUTPUT\_CFG\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 音频输出设备类型 详见 [NETDEV\_AUDIO\_CHL\_MODE\_E](#_音频输入模式枚举) |
| udwGain | 音频输出增益 范围[0,255] 不支持音频输出设备类型，即AudioOutTypeNum为0时，此字段可选 |
| byRes | 保留字段 |

### 音频文件信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwID | 音频文件ID |
| udwStatus | 音频文件状态 0：未启用 1：启用 |
| udwInstallType | 置式 0：内置 1：外置 |
| szFileName | 音频文件名 最长256个字节 |
| byRes | 保留字段 |

### 通道信息列表

|  |
| --- |
| typedef struct tagNETDEVChannelsInfos  {  UINT32 udwNum;  UINT32 audwChannelList[NETDEV\_CHANNEL\_MAX];  BYTE byRes[128];  }NETDEV\_CHANNEL\_LIST\_S, \*LPNETDEV\_CHANNEL\_LIST\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwNum | 接入的通道数, 不为0 |
| audwChannelList | 接入的通道号列表 |
| byRes | 保留字段 |

### 条件查询通道信息结构体

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| dwLimit | 每次查询的数量 |
| dwOffset | 从当前序号开始查询 |
| dwQryInfoNum | 查询条件数量 |
| pstQueryInfo | 查询条件，需动态分配内存 |
| dwRecursion | 递归查询类型 0：不递归 1：向上递归 2：向下递归 |
| byRes | 保留字段 |

### 图片查询条件

|  |
| --- |
| typedef struct tagNETDEVPicQueryCond  {  UINT32 udwSearchID;  UINT32 udwLimit;  UINT32 udwOffset;  BYTE byRes[128];  }NETDEV\_PIC\_QUERY\_COND\_S,\*LPNETDEV\_PIC\_QUERY\_COND\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwSearchID | 业务号 |
| udwLimit | 每次查询的数量 |
| udwOffset | 从当前序号开始查询，序号从0开始 |
| byRes | 保留字段 |

### 图片查询结果

|  |
| --- |
| typedef struct tagNETDEVPicQueryResult  {  UINT32 udwTotal;  UINT32 udwNum;  BYTE byRes[128];  }NETDEV\_PIC\_QUERY\_RESULT\_S,\*LPNETDEV\_PIC\_QUERY\_RESULT\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTotal | 查询结果总个数 |
| udwNum | 当前返回数量 |
| byRes | 保留字段 |

### 目标结果信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwRecordID | 记录ID |
| udwType | 记录类型 详见 [NETDEV\_OBJECT\_RECORD\_TYPE\_E](#_目标记录类型) |
| udwTime | 记录时间，UTC格式，单位秒 |
| szChannelName | 抓拍通道名称，范围[1,63] |
| udwChannelID | 抓拍通道ID |
| stBigImageInfo | 大图信息 |
| udwObjectInfoNum | 目标信息列表数量 |
| pstObjectInfoList | 目标信息列表 需动态申请 |
| byRes | 保留字段 |

### 目标信息列表

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwRecordID | 记录ID |
| udwType | 记录类型 详见 [NETDEV\_OBJECT\_RECORD\_TYPE\_E](#_目标记录类型) |
| udwTime | 记录时间，UTC格式，单位秒 |
| szChannelName | 抓拍通道名称，范围[1,63] |
| udwChannelID | 抓拍通道ID |
| stBigImageInfo | 大图信息 |
| udwObjectInfoNum | 目标信息列表数量 |
| pstObjectInfoList | 目标信息列表 需动态申请 |
| byRes | 保留字段 |

### 告警关联数据

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stStructDataInfo | 告警关联结构化数据信息 |
| byRes | 保留字段 |

### 图片告警基础信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szReference | 描述信息 |
| udwAlarmType | 告警类型，参见枚举 [NETDEV\_ALARM\_TYPE\_E](#_告警类型枚举_1) |
| tTimeStamp | 告警时间 从1970年1月1日0点开始的秒数 |
| udwSeq | 告警序号 |
| udwSourceID | 告警源ID VMS和IPC使用 |
| szSourceName | 告警源名称，长度[1,63] |
| szDeviceID | 设备ID，国表协议接入时填写国标注册码，长度[1,32] IPC、VM平台支持 |
| szRelatedID | 图片数据和告警事件的关联ID，同一个相机内全局唯一，长度为15个字符 |
| byRes | 保留字段 |

### 告警图片数据

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stAlarmPicBaseInfo | 图片告警基础信息 |
| udwImageNum | 图像个数 |
| pstImageInfo | 图像相关信息 需动态申请内存 |
| byRes | 保留字段 |

### 人数统计告警信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwType | 人数统计告警类型，参见枚举 [NETDEV\_PEOPLE\_COUNT\_ALARM\_TYPE\_E](#_人数统计告警类型) |
| stAreaRuleInfo | 人数统计区域规则统计数据信息 |
| stLineRuleInfo | 人数统计绊线规则统计数据信息 |
| stCrowdDensityInfo | 人员密度统计信息 |
| byRes | 保留字段 |

### 人数统计区域规则统计数据信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szReference | 描述信息 |
| tTimeStamp | 告警时间 从1970年1月1日0点开始的秒数 |
| udwSeq | 告警序号 |
| szDeviceID | 相机编码或域编码，回传事件订阅下发的设备编码，当事件订阅接口中携带设备编码时必填 |
| udwChannelID | 通道号，从0开始 |
| udwAreaNum | 区域规则数量，从0开始，0代表无区域规则上报 |
| pstAreaRuleCountDataList | 区域规则统计数据, 需动态申请内存 |
| byRes | 保留字段 |

### 人数统计绊线规则统计数据信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szReference | 描述信息 |
| tTimeStamp | 上报时间 从1970年1月1日0点开始的秒数 |
| udwSeq | 推送数据序号 |
| szDeviceID | 相机编码或域编码，回传事件订阅下发的设备编码，当事件订阅接口中携带设备编码时必填 |
| udwChannelID | 通道号，从0开始 |
| udwLineNum | 伴线规则数量，从0开始，0代表无区域规则上报 |
| pstLineRuleCountDataList | 伴线规则统计数据信息 需动态申请内存 |
| byRes | 保留字段 |

### 人员密度统计信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| szReference | 订阅者描述信息 |
| tTimeStamp | 告警时间 从1970年1月1日0点开始的秒数 |
| udwSeq | 推送数据序号 |
| udwSrcID | 源ID |
| szSrcName | 源名称,长度[1,63] |
| udwRelatedID | 告警事件关联ID |
| stCrowdDensityData | 人员密度统计数据 |
| byRes | 保留字段 |

### 区域规则统计数据

|  |
| --- |
| typedef struct tagNETDEVAreaRuleCountData  {  UINT32 udwAreaID;  UINT32 udwObjectNum;  BYTE byRes[256];  }NETDEV\_AREA\_RULE\_COUNT\_DATA\_S,\*LPNETDEV\_AREA\_RULE\_COUNT\_DATA\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwAreaID | 检测区域ID号，从0开始 |
| udwObjectNum | 区域规则当前人数 |
| byRes | 保留字段 |

### 伴线规则统计数据

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwLineID | 伴线ID号，从0开始 |
| szBeginPassTime | 检测开始时间：YYYYMMDDHHMMSS，时间按24小时制。字符串长度[0,18] |
| szEndPassTime | 检测结束时间：YYYYMMDDHHMMSS，时间按24小时制。字符串长度[0,18] |
| udwObjectIn | 配置时间内进入人数 |
| udwObjectOut | 配置时间内离开人数 |
| byRes | 保留字段 |

### 人员密度统计数据

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwGroupID | 统计组ID |
| tBeginTime | 统计开始时间，UTC时间 |
| tEndTime | 统计结束时间，UTC时间 |
| udwObjectIn | 进入人数 |
| udwObjectOut | 离开人数 |
| stCrowdDensitySummaryInfo | 统计汇总信息 |
| byRes | 保留字段 |

### 人员密度统计汇总信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwTotalIn | 总进入人数 |
| udwTotalOut | 总离开人数 |
| udwAlarmThermal | 报警阈值人数 |
| byRes | 保留字段 |

### 人员密度报警统计组信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwGroupID | 统计组ID |
| szGroupName | 统计组名称，长度范围[0,63] |
| udwStatisticalType | 统计类型 参见# [NETDEV\_CROWD\_DENSITY\_STATISTIC\_TYPE\_E](#_人员密度报警统计类型) |
| udwChannelRuleNum | 通道规则个数 |
| pstChannelRuleInfoList | 通道规则信息列表，通道规则个数为0时可选，需用户分配内存 |
| stCrowdDensityRuleInfo | 滞留规则 |
| udwReportInterval | 上报间隔时间 单位:(秒) |
| byRes | 保留字段 |

### 通道规则信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| udwChannelID | 通道ID |
| udwRuleNum | 规则个数 |
| audwRuleIDList | 规则ID列表 |
| byRes | 保留字段 |

### 滞留规则信息

|  |
| --- |
| 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：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| stMinorAlarmRuleInfo | 轻度报警规则 |
| stMajorAlarmRuleInfo | 中度报警规则 |
| stCriticalAlarmRuleInfo | 严重报警规则 |
| byRes | 保留字段 |

### 报警规则信息

|  |
| --- |
| typedef struct tagNETDEVAlarmRuleInfo  {  BOOL bEnabled;  UINT32 udwAlarmThermal;  BYTE byRes[128];  }NETDEV\_ALARM\_RULE\_INFO\_S,\*LPNETDEV\_ALARM\_RULE\_INFO\_S; |

**Members：**

|  |  |
| --- | --- |
| 参数 | 说明 |
| bEnabled | 是否使能, FALSE:不使能 TRUE:使能 |
| udwAlarmThermal | 报警人数阈值 |
| byRes | 保留字段 |

## 枚举定义

### 接入协议枚举

typedef enum tagNETDEVLoginProto

{

NETDEV\_LOGIN\_PROTO\_ONVIF = 0, /\* 以ONVIF协议接入\*/

NETDEV\_LOGIN\_PROTO\_PRIVATE = 1 /\* 以私有协议接入\*/

}NETDEV\_LOGIN\_PROTO\_E;

### 异常回调的消息类型枚举

typedef enum tagNETDEVException

{

NETDEV\_EXCEPTION\_REPORT\_REMUXING\_FINISH = 284, /\* 转封装完成 \*/

NETDEV\_EXCEPTION\_REPORT\_VOD\_END = 300, /\* 回放结束 \*/

NETDEV\_EXCEPTION\_REPORT\_VOD\_ABEND = 301, /\* 回放异常 \*/

NETDEV\_EXCEPTION\_REPORT\_BACKUP\_END = 302, /\* 备份结束 \*/

NETDEV\_EXCEPTION\_REPORT\_BACKUP\_DISC\_OUT = 303, /\* 磁盘被拔出 \*/

NETDEV\_EXCEPTION\_REPORT\_BACKUP\_DISC\_FULL = 304, /\* 磁盘已满 \*/

NETDEV\_EXCEPTION\_REPORT\_BACKUP\_ABEND = 305, /\* 其他原因导致备份失败 \*/

NETDEV\_EXCEPTION\_EXCHANGE = 0x8000, /\* 用户交互时异常（保活超时）\*/

NETDEV\_EXCEPTION\_REPORT\_ALARM\_INTERRUPT = 0x8001, /\* 告警上报异常结束 保活失败或者长连接断开\*/

NETDEV\_EXCEPTION\_REPORT\_MAX, /\* 最大值 \*/

NETDEV\_EXCEPTION\_REPORT\_NOT\_VALID\_PERIOD, /\* 不在有效期内 \*/

NETDEV\_EXCEPTION\_REPORT\_NOT\_VALID\_TIME, /\* 不在有效时段内 \*/

NETDEV\_EXCEPTION\_REPORT\_INVALID = 0xFFFF /\* 无效值 \*/

}NETDEV\_EXCEPTION\_TYPE\_E;

### 设备类型枚举

typedef enum tagNETDEVDeviceType

{

NETDEV\_DTYPE\_UNKNOWN = 0, /\* Unknown type \*/

NETDEV\_DTYPE\_IPC = 1, /\* IPC range \*/

NETDEV\_DTYPE\_IPC\_FISHEYE = 2, /\* 非经济型鱼眼 \*/

NETDEV\_DTYPE\_IPC\_ECONOMIC\_FISHEYE = 3, /\* 经济型鱼眼 \*/

NETDEV\_DTYPE\_IPC\_ACS = 4, /\* 人脸门禁 \*/

NETDEV\_DTYPE\_NVR = 101, /\* NVR range \*/

NETDEV\_DTYPE\_NVR\_BACKUP = 102, /\* NVR备份服务器\*/

NETDEV\_DTYPE\_HNVR = 103, /\* 混合NVR \*/

NETDEV\_DTYPE\_DC = 201, /\* DC range \*/

NETDEV\_DTYPE\_DC\_ADU = 202, /\* ADU range \*/

NETDEV\_DTYPE\_EC = 301, /\* EC range \*/

NETDEV\_DTYPE\_VMS = 501, /\* VMS range \*/

NETDEV\_DTYPE\_FG = 601, /\* FG range \*/

NETDEV\_DTYPE\_IPM = 701, /\* IPM range \*/

NETDEV\_DTYPE\_EDU = 801, /\* EDU range \*/

NETDEV\_DTYPE\_INVALID = 0xFFFF /\* 无效值 \*/

}NETDEV\_DEVICE\_TYPE\_E;

### 码流类型枚举

typedef enum tagNETDEVLiveStreamIndex

{

NETDEV\_LIVE\_STREAM\_INDEX\_MAIN = 0, /\* 主流 Main stream \*/

NETDEV\_LIVE\_STREAM\_INDEX\_AUX = 1, /\* 辅流 Sub stream \*/

NETDEV\_LIVE\_STREAM\_INDEX\_THIRD = 2, /\* 第三流 Third stream \*/

NETDEV\_LIVE\_STREAM\_INDEX\_INVALID = 0xFF /\* 无效值 Invalid value \*/

}NETDEV\_LIVE\_STREAM\_INDEX\_E;

### 媒体传输协议枚举

typedef enum tagNETDEVProtocal

{

NETDEV\_TRANSPROTOCAL\_RTPUDP = 0, /\* UDP \*/

NETDEV\_TRANSPROTOCAL\_RTPTCP = 1 /\* TCP \*/

}NETDEV\_PROTOCAL\_E;

### 图像播放流畅性枚举

typedef enum tagNetDEVPictureFluency

{

NETDEV\_PICTURE\_REAL = 0, /\* 实时性优先 \*/

NETDEV\_PICTURE\_FLUENCY = 1, /\* 流畅性优先 \*/

NETDEV\_PICTURE\_BALANCE\_NEW = 3, /\* 均衡 \*/

NETDEV\_PICTURE\_RTMP\_FLUENCY = 4, /\* RTMP流畅性优先 \*/

NETDEV\_PICTURE\_USER\_DEFINED = 5, /\* 自定义-缓冲帧数可配 \*/

NETDEV\_PICTURE\_NETADJUST = 6, /\* 网络抖动自适应模式\*/

NETDEV\_PICTURE\_FLUENCY\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_PICTURE\_FLUENCY\_E;

### 流模式枚举

typedef enum tagNETDEVStreamMode

{

NETDEV\_STREAM\_MODE\_ALL = 0x0000, /\* 音频 + 视频 \*/

NETDEV\_STREAM\_MODE\_VIDEO = 0x8000 /\* 仅视频 \*/

}NETDEV\_STREAM\_MODE\_E;

### 传输类型枚举

typedef enum tagNETDEVTransType

{

NETDEV\_TRANS\_TYPE\_FORWORD = 0, /\* 一体机转发 \*/

NETDEV\_TRANS\_TYPE\_STRAIGHT = 1 /\* 下行设备直连\*/

}NETDEV\_TRANS\_TYPE\_E;

### 起流协议枚举

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;

### 媒体文件格式枚举

typedef enum tagNETDEVMediaFileFormat

{

NETDEV\_MEDIA\_FILE\_MP4 = 0, /\* MP4格式的媒体文件(音频+视频) \*/

NETDEV\_MEDIA\_FILE\_TS = 1, /\* TS格式的媒体文件(音频+视频) \*/

NETDEV\_MEDIA\_FILE\_MP4\_ADD\_TIME = 2, /\* MP4格式的媒体文件(音频+视频) ,文件名包含时间戳 \*/

NETDEV\_MEDIA\_FILE\_TS\_ADD\_TIME = 3, /\* TS格式的媒体文件(音频+视频) ,文件名包含时间戳 \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_TS = 4, /\* TS格式的音频文件(仅音频) \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_MP4 = 5, /\* MP4格式的音频文件(仅音频) \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_TS\_ADD\_TIME = 6, /\* TS格式的音频文件(仅音频),文件名包含时间戳 \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_MP4\_ADD\_TIME = 7, /\* MP4格式的音频文件(仅音频),文件名包含时间戳 \*/

NETDEV\_MEDIA\_FILE\_MP4\_ADD\_RCD\_TIME = 8, /\* MP4格式的媒体文件(音频+视频)，文件名包含开始结束时间 \*/

NETDEV\_MEDIA\_FILE\_TS\_ADD\_RCD\_TIME = 9, /\* TS格式的媒体文件(音频+视频)，文件名包含开始结束时间 \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_MP4\_ADD\_RCD\_TIME = 10, /\* MP4格式的音频文件(仅音频)，文件名包含开始结束时间 \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_TS\_ADD\_RCD\_TIME = 11, /\* TS格式的音频文件(仅音频)，文件名包含开始结束时间 \*/

NETDEV\_MEDIA\_FILE\_VIDEO\_MP4\_ADD\_RCD\_TIME = 12, /\* MP4格式的媒体文件(仅视频) \*/

NETDEV\_MEDIA\_FILE\_VIDEO\_TS\_ADD\_RCD\_TIME = 13, /\* TS格式的媒体文件(仅视频) \*/

NETDEV\_MEDIA\_FILE\_AVI = 14, /\* AVI格式的媒体文件(音频+视频) \*/

NETDEV\_MEDIA\_FILE\_AVI\_ADD\_TIME = 15, /\* AVI格式的媒体文件(音频+视频)，文件名包含开始结束时间 \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_AVI = 16, /\* AVI格式的音频文件(仅音频) \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_AVI\_ADD\_TIME = 17, /\* AVI(仅音频),文件名包含时间戳 \*/

NETDEV\_MEDIA\_FILE\_AVI\_ADD\_RCD\_TIME = 18, /\* AVI格式的音频文件(音频+视频)，文件名包含开始结束时间 \*/

NETDEV\_MEDIA\_FILE\_AUDIO\_AVI\_ADD\_RCD\_TIME = 19, /\* AVI格式的音频文件(仅音频)，文件名包含开始结束时间 \*/

NETDEV\_MEDIA\_FILE\_VIDEO\_AVI\_ADD\_RCD\_TIME = 20, /\* AVI格式的媒体文件(仅视频) \*/

NETDEV\_MEDIA\_FILE\_INVALID

}NETDEV\_MEDIA\_FILE\_FORMAT\_E;

### 抓图图片格式枚举

typedef enum tagNETDEVPictureFormat

{

NETDEV\_PICTURE\_BMP = 0, /\* 图片格式为bmp格式 \*/

NETDEV\_PICTURE\_JPG = 1, /\* 图片格式为jpg格式 \*/

NETDEV\_PICTURE\_INVALID

}NETDEV\_PICTURE\_FORMAT\_E;

### 音频格式枚举

typedef enum tagNetDEVWaveFormatInfo

{

NETDEV\_WAVE\_FORMAT\_PCM8M16 = 0, /\* PCM,8K采样率,单声道,16位采样\*/

NETDEV\_WAVE\_FORMAT\_PCM16M16 = 1, /\* PCM,16K采样率,单声道,16位采样\*/

NETDEV\_WAVE\_FORMAT\_PCM441M16 = 2, /\* PCM,44.1K采样率,单声道,16位采样\*/

NETDEV\_WAVE\_FORMAT\_PCM11025M16 = 3, /\* PCM,11.025K采样率,单声道,16位采样\*/

NETDEV\_WAVE\_FORMAT\_PCM8\_RIGHT = 4, /\* PCM，8K采样率，右声道\*/

NETDEV\_WAVE\_FORMAT\_PCM\_NORMAL = 5, /\* PCM，通用模式\*/

NETDEV\_WAVE\_FORMAT\_PCM48000M16 = 6, /\* PCM，48.0K采样率，单声道（M），16位采样\*/

NETDEV\_WAVE\_FORMAT\_UNKNOWN

}NETDEV\_WAVE\_FORMAT\_INFO\_E;

### 音频位宽枚举

typedef enum tagNetDevAudioSampleFormatType

{

NETDEV\_AUDIO\_SAMPLE\_FMT\_NONE = -1,

NETDEV\_AUDIO\_SAMPLE\_FMT\_U8 = 0, /\* 无符号位整型\*/

NETDEV\_AUDIO\_SAMPLE\_FMT\_S16 = 1, /\* 有符号位整型\*/

NETDEV\_AUDIO\_SAMPLE\_FMT\_S32 = 2, /\* 有符号位整型\*/

NETDEV\_AUDIO\_SAMPLE\_FMT\_FLT = 3, /\* 浮点型\*/

NETDEV\_AUDIO\_SAMPLE\_FMT\_DBL = 4 /\* double型\*/

}NETDEV\_AUDIO\_SAMPLE\_FORMAT\_E;

### 媒体数据流格式枚举

typedef enum tagNETDEVMediaDataFormat

{

NETDEV\_MEDIA\_DATA\_TS = 0, /\* TS \*/

NETDEV\_MEDIA\_DATA\_RTP\_ES = 3 /\* RTP + ES \*/

}NETDEV\_MEDIA\_DATA\_FORMAT\_E;

### 音频编码格式枚举

typedef enum tagNetDEVAudioEncodeType

{

NETDEV\_AUDIO\_ENCODE\_G711U = 0, /\* 音频符合G.711U律标准 \*/

NETDEV\_AUDIO\_ENCODE\_MP2 = 1, /\* 音频符合MP2编码标准 \*/

NETDEV\_AUDIO\_ENCODE\_AAC = 2, /\* 音频符合AAC编码标准 \*/

NETDEV\_AUDIO\_ENCODE\_G711A = 3, /\* 音频符合G.711A律标准 \*/

NETDEV\_AUDIO\_ENCODE\_PCM = 4, /\* 音频符合PCM律标准 \*/

NETDEV\_AUDIO\_ENCODE\_NULL = 5 /\* 无音频 \*/

}NETDEV\_AUDIO\_ENCODE\_TYPE\_E;

### 视频帧类型枚举

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;

### 视频编码格式枚举

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;

### 元数据显示类型枚举

typedef enum tagNETDEVIVAShowRule

{

NETDEV\_IVA\_SHOW\_RULE = 0x00000001, /\* 显示规则线框 \*/

NETDEV\_IVA\_SHOW\_RESULT\_TOUTH\_RULE = 0x00000002, /\* 显示触发规则目标框 \*/

NETDEV\_IVA\_SHOW\_RESULT\_UNTOUTH\_RULE = 0x00000004, /\* 显示未触发规则目标框 \*/

NETDEV\_IVA\_SHOW\_TRACK = 0x00000008 /\* 显示目标框轨迹 \*/

}NETDEV\_IVA\_SHOW\_RULE\_E;

### 视频显示比例枚举

typedef enum tagNETDEVRenderScale

{

NETDEV\_RENDER\_SCALE\_FULL = 0, /\* 图像填充整个播放窗格 \*/

NETDEV\_RENDER\_SCALE\_PROPORTION = 1, /\* 图像按比例显示 \*/

NETDEV\_RENDER\_SCALE\_INVALID = 0xFF

}NETDEV\_RENDER\_SCALE\_E;

### 录像存储类型枚举

typedef enum tagNETDEVPLanStoreType

{

NETDEV\_STORE\_TYPE\_COMMON = 0, /\* 常规存储 \*/

NETDEV\_STORE\_TYPE\_DIGITIALINPUT = 1, /\* 开关量输入告警存储 \*/

NETDEV\_STORE\_TYPE\_MANUL = 2, /\* 手动存储 \*/

NETDEV\_STORE\_TYPE\_AUDIODETECT = 3, /\* 音频检测告警存储 \*/

NETDEV\_STORE\_TYPE\_MOTION = 4, /\* 运动检测告警存储 \*/

NETDEV\_STORE\_TYPE\_DIGITALINPUT = 5, /\* 数字输入事件存储 \*/

NETDEV\_STORE\_TYPE\_FACEDETECT = 6, /\* 人脸检测告警存储 \*/

NETDEV\_STORE\_TYPE\_VIDEO\_LOSS = 7, /\* 视频丢失存储 \*/

NETDEV\_STORE\_TYPE\_LINEDETECT = 8, /\* 越界检测告警存储 \*/

NETDEV\_STORE\_TYPE\_FIELDDETECT = 9, /\* 区域入侵检测告警存储 \*/

NETDEV\_STORE\_TYPE\_FOCUSDETECT = 10, /\* 图像虚焦告警存储 \*/

NETDEV\_STORE\_TYPE\_SCENECHANGE = 11, /\* 场景变更告警存储 \*/

NETDEV\_STORE\_TYPE\_ALARM = 12, /\* 告警 \*/

NETDEV\_STORE\_TYPE\_ALARM\_AND\_MOTION = 13, /\* 运动检测 和 告警 \*/

NETDEV\_STORE\_TYPE\_ALARM\_OR\_MOTION = 14, /\* 运动检测 或 告警 \*/

NETDEV\_STORE\_TYPE\_CAMERA\_DISCONNECT = 15, /\* 监控点断线 \*/

NETDEV\_STORE\_TYPE\_THIRD\_STREAM = 16, /\* 第三流存储 \*/

NETDEV\_STORE\_TYPE\_EVENT\_ALL\_ALARM = 17, /\* 事件类型录像，包涵所有告警类型 \*/

NETDEV\_STORE\_TYPE\_EVENT\_ALL\_TYPE = 18, /\* 事件类型录像，包涵所有录像类型 \*/

NETDEV\_PLAN\_STORE\_TYPE\_EVENT\_WITHOUT\_RESUME = 19, /\* 没有恢复的事件类型存储 \*/

NETDEV\_STORE\_TYPE\_SMART\_TRACK = 20, /\* 智能跟踪 \*/

NETDEV\_STORE\_TYPE\_URGENT\_BELL = 21, /\* 紧急铃 \*/

NETDEV\_STORE\_TYPE\_REMAIN\_ARTICLE = 22, /\* 物品遗留 \*/

NETDEV\_STORE\_TYPE\_MOVE\_ARTICLE = 23, /\* 物品搬移 \*/

NETDEV\_STORE\_TYPE\_SMART\_RECORD = 24, /\* 所有智能类告警 \*/

NETDEV\_PLAN\_STORE\_TYPE\_MAX = 25, /\* 存储类型最大值 \*/

NETDEV\_STORE\_TYPE\_HUMAN\_DETECT = 26, /\* 人形检测 \*/

NETDEV\_STORE\_TYPE\_ELEVATOR\_ENTRANCE\_DETECT = 27, /\* 入梯检测 \*/

NETDEV\_STORE\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_STORE\_TYPE\_E;

### 录像存储位置枚举

typedef enum tagNETDEVRecordLocation

{

NETDEV\_RECORD\_LOCATION\_ALL = 0, /\* 存储位置：所有 \*/

NETDEV\_RECORD\_LOCATION\_VMS = 1, /\* 存储位置：VMS \*/

NETDEV\_RECORD\_LOCATION\_NVR = 2, /\* 存储位置：NVR \*/

NETDEV\_RECORD\_LOCATION\_BACKUP = 3, /\* 存储位置：备份 \*/

NETDEV\_RECORD\_LOCATION\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_RECORD\_LOCATION\_E;

### 录像查询位置枚举

typedef enum tagNETDEVVideoPosition

{

NETDEV\_POSITION\_LOCAL = 1, /\* 设备自身录像 \*/

NETDEV\_POSITION\_CHANNEL = 2, /\* 设备下的通道录像 \*/

NETDEV\_POSITION\_INVALID

}NETDEV\_VIDEO\_POSITION\_E;

### 录像状态枚举

typedef enum tagNETDEVVideoStatus

{

NETDEV\_VIDEO\_NONE = 0, /\* 无录像 No video \*/

NETDEV\_VIDEO\_EVENT = 1, /\* 事件录像 Event video \*/

NETDEV\_VIDEO\_NORMAL = 2, /\* 普通录像 Normal video \*/

NETDEV\_VIDEO\_INVALID

}NETDEV\_VIDEO\_STATUS\_E;

### 录像下载速度枚举

typedef enum tagNETDEVDownloadSpeed

{

NETDEV\_DOWNLOAD\_SPEED\_ONE = 0, /\* 一倍速 \*/

NETDEV\_DOWNLOAD\_SPEED\_TWO = 1, /\* 二倍速 \*/

NETDEV\_DOWNLOAD\_SPEED\_FOUR = 2, /\* 四倍速 \*/

NETDEV\_DOWNLOAD\_SPEED\_EIGHT = 3, /\* 八倍速 \*/

NETDEV\_DOWNLOAD\_SPEED\_TWO\_IFRAME = 4, /\* 二倍速I帧 \*/

NETDEV\_DOWNLOAD\_SPEED\_FOUR\_IFRAME = 5, /\* 四倍速I帧 \*/

NETDEV\_DOWNLOAD\_SPEED\_EIGHT\_IFRAME = 6, /\* 八倍速I帧 \*/

NETDEV\_DOWNLOAD\_SPEED\_HALF = 7, /\* 1/2倍速 1/2x \*/

NETDEV\_DOWNLOAD\_SPEED\_FORTY = 8, /\* 40倍速 40x \*/

NETDEV\_DOWNLOAD\_SEPPD\_SIXTEEN = 9, /\* 十六倍速 \*/

NETDEV\_DOWNLOAD\_SPEED\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_E\_DOWNLOAD\_SPEED\_E;

### 播放/下载速度枚举

typedef enum tagNETDEVVodPlayStatus

{

NETDEV\_PLAY\_STATUS\_16\_BACKWARD = 0, /\* 16倍速后退播放 \*/

NETDEV\_PLAY\_STATUS\_8\_BACKWARD = 1, /\* 8倍速后退播放 \*/

NETDEV\_PLAY\_STATUS\_4\_BACKWARD = 2, /\* 4倍速后退播放 \*/

NETDEV\_PLAY\_STATUS\_2\_BACKWARD = 3, /\* 2倍速后退播放 \*/

NETDEV\_PLAY\_STATUS\_1\_BACKWARD = 4, /\* 正常速度后退播放 \*/

NETDEV\_PLAY\_STATUS\_HALF\_BACKWARD = 5, /\* 1/2倍速后退播放 \*/

NETDEV\_PLAY\_STATUS\_QUARTER\_BACKWARD = 6, /\* 1/4倍速后退播放 \*/

NETDEV\_PLAY\_STATUS\_QUARTER\_FORWARD = 7, /\* 1/4倍速播放 \*/

NETDEV\_PLAY\_STATUS\_HALF\_FORWARD = 8, /\* 1/2倍速播放 \*/

NETDEV\_PLAY\_STATUS\_1\_FORWARD = 9, /\* 正常速度前进播放 \*/

NETDEV\_PLAY\_STATUS\_2\_FORWARD = 10, /\* 2倍速前进播放 \*/

NETDEV\_PLAY\_STATUS\_4\_FORWARD = 11, /\* 4倍速前进播放 \*/

NETDEV\_PLAY\_STATUS\_8\_FORWARD = 12, /\* 8倍速前进播放 \*/

NETDEV\_PLAY\_STATUS\_16\_FORWARD = 13, /\* 16倍速前进播放 \*/

NETDEV\_PLAY\_STATUS\_2\_FORWARD\_IFRAME = 14, /\* 2倍速前进播放(I帧) \*/

NETDEV\_PLAY\_STATUS\_4\_FORWARD\_IFRAME = 15, /\* 4倍速前进播放(I帧) \*/

NETDEV\_PLAY\_STATUS\_8\_FORWARD\_IFRAME = 16, /\* 8倍速前进播放(I帧) \*/

NETDEV\_PLAY\_STATUS\_16\_FORWARD\_IFRAME = 17, /\* 16倍速前进播放(I帧) \*/

NETDEV\_PLAY\_STATUS\_2\_BACKWARD\_IFRAME = 18, /\* 2倍速后退播放(I帧) \*/

NETDEV\_PLAY\_STATUS\_4\_BACKWARD\_IFRAME = 19, /\* 4倍速后退播放(I帧) \*/

NETDEV\_PLAY\_STATUS\_8\_BACKWARD\_IFRAME = 20, /\* 8倍速后退播放(I帧) \*/

NETDEV\_PLAY\_STATUS\_16\_BACKWARD\_IFRAME = 21, /\* 16倍速后退播放(I帧) \*/

NETDEV\_PLAY\_STATUS\_INTELLIGENT\_FORWARD = 22, /\* 智能播放 \*/

NETDEV\_PLAY\_STATUS\_1\_FRAME\_FORWD = 23, /\* 单帧前进播放 \*/

NETDEV\_PLAY\_STATUS\_1\_FRAME\_BACK = 24, /\* 单帧后退播放 \*/

NETDEV\_PLAY\_STATUS\_40\_FORWARD = 25, /\* 40倍速前进播放 \*/

NETDEV\_PLAY\_STATUS\_32\_FORWARD\_IFRAME = 26, /\* 32倍速抽帧前进播放 \*/

NETDEV\_PLAY\_STATUS\_32\_BACKWARD\_IFRAME = 27, /\* 32倍速抽帧后退播放 \*/

NETDEV\_PLAY\_STATUS\_64\_FORWARD\_IFRAME = 28, /\* 64倍速抽帧前进播放 \*/

NETDEV\_PLAY\_STATUS\_64\_BACKWARD\_IFRAME = 29, /\* 64倍速抽帧后退播放 \*/

NETDEV\_PLAY\_STATUS\_128\_FORWARD\_IFRAME = 30, /\* 128倍速抽帧前进播放 \*/

NETDEV\_PLAY\_STATUS\_128\_BACKWARD\_IFRAME = 31, /\* 128倍速抽帧后退播放 \*/

NETDEV\_PLAY\_STATUS\_256\_FORWARD\_IFRAME = 32, /\* 256倍速抽帧前进播放 \*/

NETDEV\_PLAY\_STATUS\_256\_BACKWARD\_IFRAME = 33, /\* 256倍速抽帧后退播放 \*/

NETDEV\_PLAY\_STATUS\_32\_FORWARD = 34, /\* 32倍速前进播放 \*/

NETDEV\_PLAY\_STATUS\_32\_BACKWARD = 35, /\* 32倍速后退播放 \*/

NETDEV\_PLAY\_STATUS\_INVALID

}NETDEV\_VOD\_PLAY\_STATUS\_E;

### 云台命令枚举

typedef enum tagNETDEVPtzCmdEnum

{

NETDEV\_PTZ\_IRISCLOSE\_STOP =0x0101, /\* 光圈关停止 \*/

NETDEV\_PTZ\_IRISCLOSE =0x0102, /\* 光圈关 \*/

NETDEV\_PTZ\_IRISOPEN\_STOP =0x0103, /\* 光圈开停止 \*/

NETDEV\_PTZ\_IRISOPEN =0x0104, /\* 光圈开 \*/

NETDEV\_PTZ\_FOCUSNEAR\_STOP =0x0201, /\* 近聚集停止 \*/

NETDEV\_PTZ\_FOCUSNEAR =0x0202, /\* 近聚集 \*/

NETDEV\_PTZ\_FOCUSFAR\_STOP =0x0203, /\* 远聚集停止 \*/

NETDEV\_PTZ\_FOCUSFAR =0x0204, /\* 远聚集 \*/

NETDEV\_PTZ\_ZOOMTELE\_STOP = 0x0301, /\* 放大停止 \*/

NETDEV\_PTZ\_ZOOMTELE = 0x0302, /\* 放大 \*/

NETDEV\_PTZ\_ZOOMWIDE\_STOP = 0x0303, /\* 缩小停止 \*/

NETDEV\_PTZ\_ZOOMWIDE = 0x0304, /\* 缩小 \*/

NETDEV\_PTZ\_TILTUP = 0x0402, /\* 向上 \*/

NETDEV\_PTZ\_TILTDOWN = 0x0404, /\* 向下 \*/

NETDEV\_PTZ\_PANRIGHT = 0x0502, /\* 向右 \*/

NETDEV\_PTZ\_PANLEFT = 0x0504, /\* 向左 \*/

NETDEV\_PTZ\_LEFTUP = 0x0702, /\* 左上 \*/

NETDEV\_PTZ\_LEFTDOWN = 0x0704, /\* 左下 \*/

NETDEV\_PTZ\_RIGHTUP = 0x0802, /\* 右上 \*/

NETDEV\_PTZ\_RIGHTDOWN = 0x0804, /\* 右下 \*/

NETDEV\_PTZ\_ALLSTOP = 0x0901, /\* 全停命令字 \*/

NETDEV\_PTZ\_FOCUS\_AND\_IRIS\_STOP = 0x0907, /\* 聚焦.光圈停止\*/

NETDEV\_PTZ\_MOVE\_STOP = 0x0908, /\* 移动停止命令字 \*/

NETDEV\_PTZ\_ZOOM\_STOP = 0x0909, /\* 变倍停止命令字 \*/

NETDEV\_PTZ\_TRACKCRUISE = 0x1001, /\* 开始轨迹巡航 \*/

NETDEV\_PTZ\_TRACKCRUISESTOP = 0x1002, /\* 停止轨迹巡航 \*/

NETDEV\_PTZ\_TRACKCRUISEREC = 0x1003, /\* 开始录制轨迹 \*/

NETDEV\_PTZ\_TRACKCRUISERECSTOP = 0x1004, /\* 停止录制轨迹 \*/

NETDEV\_PTZ\_TRACKCRUISEADD = 0x1005, /\* 添加巡航轨迹 \*/

NETDEV\_PTZ\_TRACKCRUISEDEL = 0x1006, /\* 删除轨迹巡航 \*/

NETDEV\_PTZ\_AREAZOOMIN = 0x1101, /\* 拉框放大\*/

NETDEV\_PTZ\_AREAZOOMOUT = 0x1102, /\* 拉框缩小 Zoom out area \*/

NETDEV\_PTZ\_AREAZOOM3D = 0x1103, /\* 3D定位 3D positioning \*/

NETDEV\_PTZ\_BRUSHON = 0x0A01, /\* 雨刷开 Wiper on \*/

NETDEV\_PTZ\_BRUSHOFF = 0x0A02, /\* 雨刷关 Wiper off \*/

NETDEV\_PTZ\_LIGHTON = 0x0B01, /\* 灯开 Lamp on \*/

NETDEV\_PTZ\_LIGHTOFF = 0x0B02, /\* 灯关 Lamp off \*/

NETDEV\_PTZ\_HEATON = 0x0C01, /\* 加热开 Heater on \*/

NETDEV\_PTZ\_HEATOFF = 0x0C02, /\* 加热关 Heater off \*/

NETDEV\_PTZ\_SNOWREMOINGON = 0x1301, /\* 除雪开 Snowremoval on \*/

NETDEV\_PTZ\_SNOWREMOINGOFF = 0x1302, /\* 除雪关 Snowremoval off \*/

NETDEV\_PTZ\_INFRAREDON = 0x0D01, /\* 红外开 IR on \*/

NETDEV\_PTZ\_INFRAREDOFF = 0x0D02, /\* 红外关 IR off \*/

NETDEV\_PTZ\_SELF\_CHECKING = 0x1801, /\* 自检:当参数dwSpeed为1时,执行云台自检;当为0时,执行恢复球机云台默认物理参数,此时会重启\*/

NETDEV\_PTZ\_INVALID

} NETDEV\_PTZ\_E;

### 云台守望配置命令

typedef enum tagNETDEVPTZGuardCmd

{

NETDEV\_PTZ\_GUARD\_CMD\_GET = 0, /\* 获取指定通道的云台守望信息\*/

NETDEV\_PTZ\_GUARD\_CMD\_SET = 1 /\* 设置指定通道的云台守望信息\*/

}NETDEV\_PTZ\_GUARD\_CMD\_E;

### 云台守望模式

typedef enum tagNETDEVPTZGuardMode

{

NETDEV\_PTZ\_GUARD\_MODE\_PRESET = 0, /\* 预置位模式\*/

NETDEV\_PTZ\_GUARD\_MODE\_CRUISE = 1, /\* 预置位巡航模式\*/

NETDEV\_PTZ\_GUARD\_MODE\_TRACK = 2 /\* 轨迹巡航模式\*/

}NETDEV\_PTZ\_GUARD\_MODE\_E;

### 云台看守位操作命令

typedef enum tagNETDEVPtzHomePositionCmd

{

NETDEV\_PTZ\_SET\_HOMEPOSITION = 0, /\* 设置看守位 Set home position \*/

NETDEV\_PTZ\_GOTO\_HOMEPOSITION = 1 /\* 转到看守位 Go to home position \*/

}NETDEV\_PTZ\_HOMEPOSITIONCMD\_E;

### 聚焦状态

typedef enum tagNETDEVPTZMoveStatus

{

NETDEV\_PTZ\_MOVE\_STATUS\_IDLE = 0, /\* 空闲Free \*/

NETDEV\_PTZ\_MOVE\_STATUS\_MOVING = 1, /\* 正在聚焦Moving \*/

NETDEV\_PTZ\_MOVE\_STATUS\_INVALID = 0xff /\* 无效值Invalid value \*/

}NETDEV\_PTZ\_MOVE\_STATUS\_E;

### 巡航状态枚举

typedef enum tagNETDEVPTZTrackStatus

{

NETDEV\_PTZ\_TRACK\_STATUS\_FREE = 0, /\* 空闲（未巡航且未录制）\*/

NETDEV\_PTZ\_TRACK\_STATUS\_RUN = 1, /\* 正在巡航\*/

NETDEV\_PTZ\_TRACK\_STATUS\_START\_RECORD = 2 /\*正在录制\*/

}NETDEV\_PTZ\_TRACK\_STATUS\_E;

### 云台巡航操作枚举

typedef enum tagNETDEVPTZTRACKCmd

{

NETDEV\_PTZ\_TRACKCRUISE\_START = 0, /\* 开始轨迹巡航 \*/

NETDEV\_PTZ\_TRACKCRUISE\_STOP = 1, /\* 停止轨迹巡航 \*/

NETDEV\_PTZ\_TRACKCRUISE\_REC = 2, /\* 开始录制轨迹 \*/

NETDEV\_PTZ\_TRACKCRUISE\_RECSTOP = 3, /\* 停止录制轨迹 \*/

NETDEV\_PTZ\_TRACKCRUISE\_ADD = 4, /\* 添加巡航轨迹，一体机服务器暂不支持该接口，轨迹巡航ID写死为16\*/

NETDEV\_PTZ\_TRACKCRUISE\_DEL = 5 /\* 删除轨迹巡航 \*/

}NETDEV\_PTZ\_TRACKCMD\_E;

### 云台预置位操作命令枚举

typedef enum tagNETDEVPtzPresetCmd

{

NETDEV\_PTZ\_SET\_PRESET = 0, /\* 设置预置位 \*/

NETDEV\_PTZ\_CLE\_PRESET = 1, /\* 清除预置位 \*/

NETDEV\_PTZ\_GOTO\_PRESET = 2 /\* 转到预置位 \*/

}NETDEV\_PTZ\_PRESETCMD\_E;

### 云台巡航操作枚举

typedef enum tagNETDEVPTZCruiseCmd

{

NETDEV\_PTZ\_ADD\_CRUISE = 0, /\* 添加巡航路径 \*/

NETDEV\_PTZ\_MODIFY\_CRUISE = 1, /\* 编辑巡航路径 \*/

NETDEV\_PTZ\_DEL\_CRUISE = 2, /\* 删除巡航路径 \*/

NETDEV\_PTZ\_RUN\_CRUISE = 3, /\* 开始巡航 \*/

NETDEV\_PTZ\_STOP\_CRUISE = 4, /\* 停止巡航 \*/

NETDEV\_PTZ\_GET\_CRUISE = 5 /\* 获取巡航路径 \*/

}NETDEV\_PTZ\_CRUISECMD\_E;

### 云台方位枚举

typedef enum tagNETDEVPTZDirection

{

NETDEV\_PTZ\_DIRECTION\_EAST = 0, /\* 东 \*/

NETDEV\_PTZ\_DIRECTION\_SOUTHEAST = 1, /\* 东南 \*/

NETDEV\_PTZ\_DIRECTION\_SOUTH = 2, /\* 南 \*/

NETDEV\_PTZ\_DIRECTION\_SOUTHWEST = 3, /\* 西南 \*/

NETDEV\_PTZ\_DIRECTION\_WEST = 4, /\* 西 \*/

NETDEV\_PTZ\_DIRECTION\_NORTHWEST = 5, /\* 西北 \*/

NETDEV\_PTZ\_DIRECTION\_NORTH = 6, /\* 北 \*/

NETDEV\_PTZ\_DIRECTION\_NORTHEAST = 7, /\* 东北 \*/

NETDEV\_PTZ\_DIRECTION\_AUTO = 8, /\* 自动 \*/

NETDEV\_PTZ\_DIRECTION\_INVALID = 0xFFFF /\* 无效值\*/

}NETDEV\_PTZ\_DIRECTION\_E;

### 回放控制命令枚举

typedef enum tagNETDEVPlayControl

{

NETDEV\_PLAY\_CTRL\_PLAY = 0, /\* 开始播放 \*/

NETDEV\_PLAY\_CTRL\_PAUSE = 1, /\* 暂停播放 \*/

NETDEV\_PLAY\_CTRL\_RESUME = 2, /\* 恢复播放 \*/

NETDEV\_PLAY\_CTRL\_GETPLAYTIME = 3, /\* 获取播放进度 \*/

NETDEV\_PLAY\_CTRL\_SETPLAYTIME = 4, /\* 设置播放进度 \*/

NETDEV\_PLAY\_CTRL\_GETPLAYSPEED = 5, /\* 获取播放速度 \*/

NETDEV\_PLAY\_CTRL\_SETPLAYSPEED = 6, /\* 设置播放速度 \*/

NETDEV\_PLAY\_CTRL\_SET\_SINGLE\_FRAME\_SPEED = 7, /\* 设置单帧播放 \*/

NETDEV\_PLAY\_CTRL\_UPDATE\_M3U8URL = 8 /\* 更新M3u8URL \*/

}NETDEV\_VOD\_PLAY\_CTRL\_E;

### 日志主类型枚举

typedef enum tagNETDEVLogMainType

{

/\* 仅NVR支持 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALL = 0, /\* 全部类型的日志 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALARM = 1, /\* 告警类 \*/

NETDEV\_LOG\_MAIN\_TYPE\_EXCEPTION = 2, /\* 异常类 \*/

NETDEV\_LOG\_MAIN\_TYPE\_OPERATION = 3, /\* 操作类 \*/

NETDEV\_LOG\_MAIN\_TYPE\_MESSAGE = 4, /\* 信息类 \*/

/\* VMS操作日志主类型\*/

NETDEV\_LOG\_MAIN\_TYPE\_VMS\_ALL = 90, /\* 全部主类型 \*/

NETDEV\_LOG\_MAIN\_TYPE\_LOGIN = 91, /\* 登录 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ORGANIZATION\_CONFIG = 100, /\* 组织配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_USER\_CONFIG = 101, /\* 用户配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ROLE\_CONFIG = 102, /\* 角色配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_DEVICE\_CONFIG = 103, /\* 设备配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_CHANNE\_LCONFIG = 104, /\* 通道配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_SERVER\_CONFIG = 105, /\* 主从服务器配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_PTZ\_CONFIG = 106, /\* 云台配置\*/

NETDEV\_LOG\_MAIN\_TYPE\_VIDEOWALL\_CONFIG = 107, /\* 电视墙配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_EMAP\_CONFIG = 108, /\* 地图配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_SYSTEM\_CONFIG = 109, /\* 系统配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_SEQUENCE\_PLAN\_CONFIG = 110, /\* 轮巡预案配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_NETWORK\_KEYPAD\_CONFIG = 111, /\* 网络键盘通道配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_SEQUENCE\_RESOURCE\_CONFIG = 112, /\* 轮巡资源配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_HOT\_SPOT\_AND\_ZONE\_CONFIG = 113, /\* 热点、热区配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_USER\_TIME\_TEMPLATE\_CONFIG = 114, /\* 用户时间模板配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_SERVER\_WORK\_MODE = 115, /\* 服务器工作模式配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_PERMISSION\_TIME\_TEMPLATE\_CONFIG =116, /\* 权限时间模板配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_TIME\_CONFIG = 117, /\* 手动时间配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_TIMESYNC\_CONFIG = 118, /\* 时间同步 \*/

NETDEV\_LOG\_MAIN\_TYPE\_HOLIDAY\_CONFIG = 119, /\* 假日启动时间 \*/

NETDEV\_LOG\_MAIN\_TYPE\_NTPTIME\_CONFIG = 120, /\* NTP时间配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_DST\_CONFIG = 121, /\* 夏令时配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_RESRELATION\_CONFIG = 122, /\* 关联资源配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_DEVICE\_TIME\_SYNC = 123, /\* 设备时间同步 \*/

NETDEV\_LOG\_MAIN\_TYPE\_TCPIP\_CONFIG = 124, /\* tcp/ip配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_EZCLOUD\_CONFIG = 125, /\* 私有云配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_DDNS\_CONFIG = 126, /\* DDNS配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_PORT\_CONFIG = 127, /\* 端口配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_PORTMAPPING\_CONFIG = 128, /\* 端口映射 \*/

NETDEV\_LOG\_MAIN\_TYPE\_EMAIL\_CONFIG = 129, /\* 邮件配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_GBSERVER\_CONFIG = 130, /\* 国标服务器配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_GBT28181\_LOCAL\_CONFIG = 131, /\* 本地服务器配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_UNP\_CLIENT\_CONFIG = 132, /\* UNP客户端配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_UNP\_SERVER\_CONFIG = 133, /\* UNP服务器配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_STATIC\_ROUTE\_CONFIG = 134, /\* 静态路由操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_802DOT1X\_CONFIG = 135, /\* 802Dot1x配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ARP\_CONFIG = 136, /\* ARP配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_HTTPS\_CONFIG = 137, /\* HTTPS配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_TELNET\_CONFIG = 138, /\* Telnet配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_SECURITY\_PSW\_CONFIG = 139, /\* 安全密码配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_IP\_FILTER\_RULE\_CONFIG = 140, /\* IP过滤规则配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_MAINTENANCE\_CONFIG = 141, /\* 系统维护 \*/

NETDEV\_LOG\_MAIN\_TYPE\_SET\_MAX\_LOG\_RETENTION\_TIME = 142, /\* 日志清理天数配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_STREAM\_TRANSMISSION\_POLICY\_CONFIG = 143, /\* 媒体流传输策略配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_BATCH\_CONFIG = 144, /\* 批量配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_RAIDMODE\_CONFIG = 145, /\* RAID模式配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ARRAY\_CONFIG = 146, /\* 阵列配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_DISK\_MANAGEMENT\_CONFIG = 147, /\* 硬盘管理 \*/

NETDEV\_LOG\_MAIN\_TYPE\_CHANNEL\_SPACE\_CONFIG = 148, /\* 通道容量配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_OVER\_WRITE\_POLICY\_CONFIG = 149, /\* 满策略配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_DISKTEST\_CONFIG = 150, /\* 硬盘检测配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_NET\_DISK\_CONFIG = 151, /\* 网络硬盘配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_RECORDING\_SCHEDULE\_CONFIG = 152, /\* 录像计划配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_RECORDING\_TIME\_TEMPLATE\_CONFIG = 153, /\* 录像时间模板配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALARM\_CONFIG = 154, /\* 告警配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALARM\_TIMETEMPLATE\_CONFIG = 155, /\* 告警时间模板配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_CONTACT\_CONFIG = 156, /\* 联系人配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALARM\_TO\_VIDEO\_WALL\_CONFIG = 157, /\* 告警联动上墙配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_VIDEO\_WALL\_SCENE\_CONFIG = 158, /\* 电视墙场景配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_VIDEO\_WALL\_WINDOW\_CONFIG = 159, /\* 电视墙窗口配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_VIDEO\_WALL\_VIRTUALLED\_CONFIG = 160, /\* 电视墙虚拟LED配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_EXTERNAL\_ALARM\_CONFIG = 161, /\* 紧急铃配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_DISK\_GROUP\_CONFIG = 162, /\* 盘组配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_BK\_PLAN\_CONFIG = 163, /\* 备份计划 \*/

NETDEV\_LOG\_MAIN\_TYPE\_BK\_TASK\_CONFIG = 164, /\* 备份任务 \*/

NETDEV\_LOG\_MAIN\_TYPE\_LOCAL\_BK\_TASKCONFIG = 165, /\* 本地备份任务 \*/

NETDEV\_LOG\_MAIN\_TYPE\_TVWALL\_AUDIO\_CONFIG = 166, /\* 电视墙音频输出配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_LIVEVIEW = 167, /\* 实况预览操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_PLAYBACK = 168, /\* 录像回放操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_TWO\_WAY\_AUDIO = 169, /\* 语音操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_PTZ = 170, /\* 云台操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_DOWNLOAD = 171, /\* 下载操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_LIVEVIEW\_ON\_VIDEO\_WALL = 172, /\* 实况上墙操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_PLAYBACK\_ON\_VIDEO\_WALL = 173, /\* 回放上墙操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_EMAP\_OPERATION = 174, /\* 电子地图操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_DC\_SEQUENCE\_OPERATION = 175, /\* DC轮巡业务操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_VIDEO\_WALL\_PLAYING\_BY\_NETWROK\_KEYPAD = 176, /\* 网络键盘操作实况上墙 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALARM\_TO\_VIDEO\_WALL = 177, /\* 报警联动上墙操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ALARM\_SUBSCRIPTION = 178, /\* 告警订阅 \*/

NETDEV\_LOG\_MAIN\_TYPE\_PRESET\_PATROL\_CONFIG = 179, /\* 预置位巡航配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_AUTOGUARD\_CONFIG = 180, /\* 守望配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_RECORDED\_PATROL\_CONFIG = 181, /\* 轨迹巡航配置 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ACCESS\_CONTROL\_PERSONNEL\_MANAGEMENT = 182, /\* 门禁人员操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ACCESS\_CONTROL\_CARD\_MANAGEMENT = 183, /\* 门禁卡操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ZONE\_OPERATION = 184, /\* 报警点操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_ACCESS\_CONTROL\_DOOR = 185, /\* 门操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_TRIGGER\_ALARMOUTPUT = 186, /\* 报警输出触发操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_CENTER\_RECORD = 187, /\* 中心录像操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_FACE\_LIBRARY\_OPERATION = 188, /\* 人脸库操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_FACE\_MEMBER\_OPERATION = 189, /\* 人脸成员操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_FACE\_CUSTOM\_OPERATION = 190, /\* 人脸自定义属性操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_FACE\_MEMBER\_SORT\_OPERATION = 191, /\* 人脸库划归操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_FACE\_MONITORING\_TASKO\_PERATION = 192, /\* 人脸布控操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_VEHICLE\_LIBRARY\_OPERATION = 193, /\* 车辆库操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_VEHICLE\_MEMBER\_SORT\_OPERATION = 194, /\* 车辆库划归操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_VEHICLE\_MEMBER\_OPERATION = 195, /\* 车辆成员操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_CAP\_PAC\_TASK = 196, /\* 抓包操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_RECORD\_BK\_CONFIG = 197, /\* 录像备份 \*/

NETDEV\_LOG\_MAIN\_TYPE\_USB\_OPERATION = 198, /\* U盘操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_VEHICLE\_MONITORING\_TASK\_OPERATION = 199, /\* 车辆布控操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_PERMISSION\_ASSIGNMENT\_OPERATION = 200, /\* 授权操作 \*/

NETDEV\_LOG\_MAIN\_TYPE\_INVALID = 0xFFFF /\* 无效值 \*/

}NETDEV\_LOG\_MAIN\_TYPE\_E;

### 日志子类型枚举

typedefenumtagNETDEVLogSubType

{

NETDEV\_LOG\_ALL\_SUB\_TYPES =0x0101, /\* 所有信息类日志 \*/

/\* 信息类日志 \*/

NETDEV\_LOG\_MSG\_HDD\_INFO =300, /\* 硬盘信息 \*/

NETDEV\_LOG\_MSG\_SMART\_INFO =301, /\* S.M.A.R.T信息 \*/

NETDEV\_LOG\_MSG\_REC\_OVERDUE =302, /\* 过期录像删除 \*/

NETDEV\_LOG\_MSG\_PIC\_REC\_OVERDUE =303, /\* 过期图片文件删除 \*/

/\* 通知类日志 \*/

NETDEV\_LOG\_NOTICE\_IPC\_ONLINE =310, /\* 设备上线 \*/

NETDEV\_LOG\_NOTICE\_IPC\_OFFLINE =311, /\* 设备下线 \*/

NETDEV\_LOG\_NOTICE\_ARRAY\_RECOVER =312, /\* 阵列恢复正常 \*/

NETDEV\_LOG\_NOTICE\_INIT\_ARRARY =313, /\* 阵列初始化 \*/

NETDEV\_LOG\_NOTICE\_REBUILD\_ARRARY =314, /\* 阵列重建 \*/

NETDEV\_LOG\_NOTICE\_POE\_PORT\_STATUS =315, /\* PoE网口状态 \*/

NETDEV\_LOG\_NOTICE\_NETWORK\_PORT\_STATUS =316, /\* 交换网口状态 \*/

NETDEV\_LOG\_NOTICE\_DISK\_ONLINE =317, /\* 磁盘上线 \*/

/\* 告警类日志的子类型日志ID \*/

NETDEV\_LOG\_ALARM\_MOTION\_DETECT =350, /\* 移动侦测告警 \*/

NETDEV\_LOG\_ALARM\_MOTION\_DETECT\_RESUME =351, /\* 移动侦测告警恢复 \*/

NETDEV\_LOG\_ALARM\_VIDEO\_LOST =352, /\* 视频丢失告警 \*/

NETDEV\_LOG\_ALARM\_VIDEO\_LOST\_RESUME =353, /\* 视频丢失告警恢复 \*/

NETDEV\_LOG\_ALARM\_VIDEO\_TAMPER\_DETECT =354, /\* 遮挡侦测告警 \*/

NETDEV\_LOG\_ALARM\_VIDEO\_TAMPER\_RESUME =355, /\* 遮挡侦测告警恢复 \*/

NETDEV\_LOG\_ALARM\_INPUT\_SW =356, /\* 输入开关量告警 \*/

NETDEV\_LOG\_ALARM\_INPUT\_SW\_RESUME =357, /\* 输入开关量告警恢复 \*/

NETDEV\_LOG\_ALARM\_IPC\_ONLINE =358, /\* 设备上线 \*/

NETDEV\_LOG\_ALARM\_IPC\_OFFLINE =359, /\* 设备下线 \*/

NETDEV\_LOG\_ALARM\_AUDIO\_DETECTION\_START =360, /\* 音频检测开始 \*/

NETDEV\_LOG\_ALARM\_AUDIO\_DETECTION\_END =361, /\* 音频检测结束 \*/

NETDEV\_LOG\_ALARM\_CROSS\_LINE\_DETECT =362, /\* 越界检测报警 \*/

NETDEV\_LOG\_ALARM\_FACE\_DETECT =363, /\* 人脸检测报警 \*/

NETDEV\_LOG\_ALARM\_INTRUSION\_DETECT =364, /\* 区域入侵报警 \*/

NETDEV\_LOG\_ALARM\_POS =365, /\* POS报警 \*/

NETDEV\_LOG\_ALARM\_IMAGETOOBLURRY\_ON =366, /\* 虚焦检测开始 \*/

NETDEV\_LOG\_ALARM\_IMAGETOOBLURRY\_OFF =367, /\* 虚焦检测结束 \*/

NETDEV\_LOG\_ALARM\_GLOBAL\_SCENE\_CHANGE =368, /\* 场景变更检测 \*/

NETDEV\_LOG\_ALARM\_AUTO\_TRACK\_ON =369, /\* 智能跟踪开始 \*/

NETDEV\_LOG\_ALARM\_AUTO\_TRACK\_OFF =370, /\* 智能跟踪结束 \*/

NETDEV\_LOG\_ALARM\_CONFLAGRATION\_ON =371, /\* 火灾告警 \*/

NETDEV\_LOG\_ALARM\_CONFLAGRATION\_OFF =372, /\* 火灾告警结束 \*/

NETDEV\_LOG\_ALARM\_HUMAN\_SHAPE\_DETECT\_ON =373, /\* 人形检测告警 \*/

NETDEV\_LOG\_ALARM\_HUMAN\_SHAPE\_DETECT\_OFF =374, /\* 人形检测告警结束 \*/

NETDEV\_LOG\_ALARM\_ENTERAREA =375, /\* 进入区域 \*/

NETDEV\_LOG\_ALARM\_LEAVEAREA =376, /\* 离开区域 \*/

/\* 异常类日志的子类型日志 \*/

NETDEV\_LOG\_EXCEP\_DISK\_ONLINE =400, /\* 磁盘上线 \*/

NETDEV\_LOG\_EXCEP\_DISK\_OFFLINE =401, /\* 磁盘下线 \*/

NETDEV\_LOG\_EXCEP\_DISK\_ERR =402, /\* 磁盘异常 \*/

NETDEV\_LOG\_EXCEP\_STOR\_ERR =403, /\* 存储错误 \*/

NETDEV\_LOG\_EXCEP\_STOR\_ERR\_RECOVER =404, /\* 存储错误恢复 \*/

NETDEV\_LOG\_EXCEP\_STOR\_DISOBEY\_PLAN =405, /\* 未按计划存储 \*/

NETDEV\_LOG\_EXCEP\_STOR\_DISOBEY\_PLAN\_RECOVER =406, /\* 未按计划存储恢复 \*/

NETDEV\_LOG\_EXCEP\_ILLEGAL\_ACCESS =407, /\* 非法访问 \*/

NETDEV\_LOG\_EXCEP\_IP\_CONFLICT =408, /\* IP地址冲突 \*/

NETDEV\_LOG\_EXCEP\_NET\_BROKEN =409, /\* 网络断开 \*/

NETDEV\_LOG\_EXCEP\_PIC\_REC\_ERR =410, /\* 抓图出错,获取图片文件失败 \*/

NETDEV\_LOG\_EXCEP\_VIDEO\_EXCEPTION =411, /\* 视频输入异常(只针对模拟通道) \*/

NETDEV\_LOG\_EXCEP\_VIDEO\_MISMATCH =412, /\* 视频制式不匹配 \*/

NETDEV\_LOG\_EXCEP\_RESO\_MISMATCH =413, /\* 编码分辨率和前端不匹配 \*/

NETDEV\_LOG\_EXCEP\_TEMP\_EXCE =414, /\* 温度异常 \*/

NETDEV\_LOG\_EXCEP\_RUNOUT\_RECORD\_SPACE =415, /\* 录像存储空间即将满 \*/

NETDEV\_LOG\_EXCEP\_RUNOUT\_IMAGE\_SPACE =416, /\* 图片存储空间即将满 \*/

NETDEV\_LOG\_EXCEP\_OUT\_RECORD\_SPACE =417, /\* 录像存储空间满 \*/

NETDEV\_LOG\_EXCEP\_OUT\_IMAGE\_SPACE =418, /\* 图片存储空间满 \*/

NETDEV\_LOG\_EXCEP\_ANRIDISASSEMBLY =419, /\* 防拆报警 \*/

NETDEV\_LOG\_EXCEP\_ANRIDISASSEMBLY\_RECOVER =420, /\* 防拆报警恢复 \*/

NETDEV\_LOG\_EXCEP\_ARRAY\_DAMAGE =421, /\* 阵列损坏 \*/

NETDEV\_LOG\_EXCEP\_ARRAY\_DEGRADE =422, /\* 阵列衰退 \*/

NETDEV\_LOG\_EXCEP\_RECORD\_SNAPSHOT\_ABNOR =423, /\* 录像/抓图异常 \*/

NETDEV\_LOG\_EXCEP\_NET\_BROKEN\_RECOVER =424, /\* 网络断开恢复 \*/

NETDEV\_LOG\_EXCEP\_IP\_CONFLICT\_RECOVER =425, /\* IP地址冲突恢复 \*/

NETDEV\_LOG\_EXCEP\_DEVICE\_HIGHTEMP =426, /\* 设备高温报警 \*/

NETDEV\_LOG\_EXCEP\_DEVICE\_LOWTEMP =427, /\* 设备低温报警 \*/

NETDEV\_LOG\_EXCEP\_DEVICE\_HIGHTEMP\_RECOVER =428, /\* 设备高温报警恢复 \*/

NETDEV\_LOG\_EXCEP\_DEVICE\_LOWTEMP\_RECOVER =429, /\* 设备低温报警恢复 \*/

NETDEV\_LOG\_EXCEP\_FAN\_FAULT =430, /\* 设备风扇故障 \*/

NETDEV\_LOG\_EXCEP\_FAN\_FAULT\_RECOVER =431, /\* 设备风扇故障恢复 \*/

NETDEV\_LOG\_EXCEP\_LEDBOX\_HIGHTEMP =432, /\* 设备电箱高温 \*/

NETDEV\_LOG\_EXCEP\_LEDBOX\_HIGHTEMP\_RECOVER =433, /\* 设备电箱高温恢复 \*/

NETDEV\_LOG\_EXCEP\_LEDBOX\_SMOKE =434, /\* 设备电箱烟雾告警 \*/

NETDEV\_LOG\_EXCEP\_LEDBOX\_SMOKE\_RECOVER =435, /\* 设备电箱烟雾告警恢复 \*/

/\* 操作类日志的子类型日志 \*/

/\* 业务类Services \*/

NETDEV\_LOG\_OPSET\_LOGIN =450, /\* 用户登录 \*/

NETDEV\_LOG\_OPSET\_LOGOUT =451, /\* 注销登录 \*/

NETDEV\_LOG\_OPSET\_USER\_ADD =452, /\* 用户添加 \*/

NETDEV\_LOG\_OPSET\_USER\_DEL =453, /\* 用户删除 \*/

NETDEV\_LOG\_OPSET\_USER\_MODIFY =454, /\* 用户修改 \*/

NETDEV\_LOG\_OPSET\_START\_REC =455, /\* 开始录像 \*/

NETDEV\_LOG\_OPSET\_STOP\_REC =456, /\* 停止录像 \*/

NETDEV\_LOG\_OPSETR\_PLAY\_DOWNLOAD =457, /\* 回放/下载 \*/

NETDEV\_LOG\_OPSET\_DOWNLOAD =458, /\* 下载 \*/

NETDEV\_LOG\_OPSET\_PTZCTRL =459, /\* 云台控制 \*/

NETDEV\_LOG\_OPSET\_PREVIEW =460, /\* 实况预览 \*/

NETDEV\_LOG\_OPSET\_REC\_TRACK\_START =461, /\* 轨迹录制开始 \*/

NETDEV\_LOG\_OPSET\_REC\_TRACK\_STOP =462, /\* 轨迹录制停止 \*/

NETDEV\_LOG\_OPSET\_START\_TALKBACK =463, /\* 开始语音对讲 \*/

NETDEV\_LOG\_OPSET\_STOP\_TALKBACK =464, /\* 停止语音对讲 \*/

NETDEV\_LOG\_OPSET\_IPC\_ADD =465, /\* 添加IPC \*/

NETDEV\_LOG\_OPSET\_IPC\_DEL =466, /\* 删除IPC \*/

NETDEV\_LOG\_OPSET\_IPC\_SET =467, /\* 设置IPC \*/

NETDEV\_LOG\_OPSET\_IPC\_QUICK\_ADD =468, /\* 快速添加IPC \*/

NETDEV\_LOG\_OPSET\_IPC\_ADD\_BY\_NETWORK =469, /\* 网段添加IPC \*/

NETDEV\_LOG\_OPSET\_IPC\_MOD\_IP =470, /\* 修改IPC网络地址 \*/

/\* 配置类 \*/

NETDEV\_LOG\_OPSET\_DEV\_BAS\_CFG =500, /\* 设备基本信息配置 \*/

NETDEV\_LOG\_OPSET\_TIME\_CFG =501, /\* 设备时间配置 \*/

NETDEV\_LOG\_OPSET\_SERIAL\_CFG =502, /\* 设备串口配置 \*/

NETDEV\_LOG\_OPSET\_CHL\_BAS\_CFG =503, /\* 通道基本配置 \*/

NETDEV\_LOG\_OPSET\_CHL\_NAME\_CFG =504, /\* 通道名称配置 \*/

NETDEV\_LOG\_OPSET\_CHL\_ENC\_VIDEO =505, /\* 视频编码参数配置 \*/

NETDEV\_LOG\_OPSET\_CHL\_DIS\_VIDEO =506, /\* 通道视频显示参数配置 \*/

NETDEV\_LOG\_OPSET\_PTZ\_CFG =507, /\* 云台配置 \*/

NETDEV\_LOG\_OPSET\_CRUISE\_CFG =508, /\* 巡航线路设置 \*/

NETDEV\_LOG\_OPSET\_PRESET\_CFG =509, /\* 预置位设置 \*/

NETDEV\_LOG\_OPSET\_VIDPLAN\_CFG =510, /\* 录像计划配置 \*/

NETDEV\_LOG\_OPSET\_MOTION\_CFG =511, /\* 运动检测配置 \*/

NETDEV\_LOG\_OPSET\_VIDLOSS\_CFG =512, /\* 视频丢失配置 \*/

NETDEV\_LOG\_OPSET\_COVER\_CFG =513, /\* 视频遮挡配置 \*/

NETDEV\_LOG\_OPSET\_MASK\_CFG =514, /\* 视频遮盖配置 \*/

NETDEV\_LOG\_OPSET\_SCREEN\_OSD\_CFG =515, /\* OSD叠加配置 \*/

NETDEV\_LOG\_OPSET\_ALARMIN\_CFG =516, /\* 报警输入配置 \*/

NETDEV\_LOG\_OPSET\_ALARMOUT\_CFG =517, /\* 报警输出配置 \*/

NETDEV\_LOG\_OPSET\_ALARMOUT\_OPEN\_MAN =518, /\* 手动开启报警输出,人机 \*/

NETDEV\_LOG\_OPSET\_ALARMOUT\_CLOSE\_MAN =519, /\* 手动关闭报警输出,人机 \*/

NETDEV\_LOG\_OPSET\_ABNORMAL\_CFG =520, /\* 异常配置 \*/

NETDEV\_LOG\_OPSET\_HDD\_CFG =521, /\* 硬盘配置 \*/

NETDEV\_LOG\_OPSET\_NET\_IP\_CFG =522, /\* TCP/IP配置 \*/

NETDEV\_LOG\_OPSET\_NET\_PPPOE\_CFG =523, /\* PPPOE配置 \*/

NETDEV\_LOG\_OPSET\_NET\_PORT\_CFG =524, /\* 端口配置 \*/

NETDEV\_LOG\_OPSET\_NET\_DDNS\_CFG =525, /\* DDNS配置 \*/

NETDEV\_LOG\_OPSET\_AUDIO\_DETECT =527, /\* 搜索扩展硬盘 \*/

NETDEV\_LOG\_OPSET\_SEARCH\_EX\_DISK =528, /\* 搜索扩展硬盘 \*/

NETDEV\_LOG\_OPSET\_ADD\_EX\_DISK =529, /\* 添加扩展硬盘 \*/

NETDEV\_LOG\_OPSET\_DEL\_EX\_DISK =530, /\* 删除扩展硬盘 \*/

NETDEV\_LOG\_OPSET\_SET\_EX\_DISK =531, /\* 配置扩展硬盘 \*/

NETDEV\_LOG\_OPSET\_LIVE\_BY\_MULTICAST =532, /\* 组播实况 \*/

NETDEV\_LOG\_OPSET\_BISC\_DEV\_INFO =533, /\* 设备基本信息配置 \*/

NETDEV\_LOG\_OPSET\_PREVIEW\_CFG =534, /\* 本地预览配置 \*/

NETDEV\_LOG\_OPSET\_SET\_EMAIL =535, /\* 邮件配置 \*/

NETDEV\_LOG\_OPSET\_TEST\_EMAIL =536, /\* 邮件测试 \*/

NETDEV\_LOG\_OPSET\_SET\_IPCONTROL =537, /\* IP权限配置 \*/

NETDEV\_LOG\_OPSET\_PORT\_MAP =538, /\* 端口映射配置 \*/

NETDEV\_LOG\_OPSET\_ADD\_TAG =539, /\* 添加录像标签 \*/

NETDEV\_LOG\_OPSET\_DEL\_TAG =540, /\* 删除录像标签 \*/

NETDEV\_LOG\_OPSET\_MOD\_TAG =541, /\* 修改录像标签 \*/

NETDEV\_LOG\_OPSET\_LOCK\_RECORD =542, /\* 录像锁定 \*/

NETDEV\_LOG\_OPSET\_UNLOCK\_RECORD =543, /\* 录像解锁定 \*/

NETDEV\_LOG\_OPSET\_DDNS\_UPDATE\_SUCCESS =545, /\* DDNS更新成功 \*/

NETDEV\_LOG\_OPSET\_DDNS\_INCORRECT\_ID =546, /\* DDNS更新失败，错误用户名密码 \*/

NETDEV\_LOG\_OPSET\_DDNS\_DOMAIN\_NAME\_NOT\_EXIST =547, /\* DDNS更新失败，域名不存在 \*/

NETDEV\_LOG\_OPSET\_DDNS\_UPDATE\_FAIL =548, /\* DDNS更新失败 \*/

NETDEV\_LOG\_OPSET\_HTTP\_CFG =549, /\* HTTPS配置 \*/

NETDEV\_LOG\_OPSET\_IP\_OFFLINE\_ALARM\_CFG =550, /\* IPC离线报警配置 \*/

NETDEV\_LOG\_OPSET\_TELNET\_CFG =551, /\* Telnet配置 \*/

NETDEV\_LOG\_OPSET\_TEST\_DDNS\_DOMAIN =552, /\* DDNS域名检测 \*/

NETDEV\_LOG\_OPSET\_DDNS\_DOMAIN\_CONFLICT =553, /\* DDNS域名冲突 \*/

NETDEV\_LOG\_OPSET\_DDNS\_DOMAIN\_INVALID =554, /\* DDNS域名不合法 \*/

NETDEV\_LOG\_OPSET\_DEL\_PRESET =555, /\* 删除预置位 \*/

NETDEV\_LOG\_OPSET\_PTZ\_3D\_POSITION =556, /\* 云台3D定位 \*/

NETDEV\_LOG\_OPSET\_SNAPSHOT\_SCHEDULE\_CFG =557, /\* 抓图计划配置 \*/

NETDEV\_LOG\_OPSET\_IMAGE\_UPLOAD\_SCHEDULE\_CFG =558, /\* 图片上传计划配置 \*/

NETDEV\_LOG\_OPSET\_FTP\_CFG =559, /\* FTP服务器配置 \*/

NETDEV\_LOG\_OPSET\_TEST\_FTP\_SERVER =560, /\* FTP服务器连接测试 \*/

NETDEV\_LOG\_OPSET\_START\_MANUAL\_SNAPSHOT =561, /\* 手动抓图开启 \*/

NETDEV\_LOG\_OPSET\_CLOSE\_MANUAL\_SNAPSHOT =562, /\* 手动抓图关闭 \*/

NETDEV\_LOG\_OPSET\_SNAPSHOT\_CFG =563, /\* 抓图参数配置 \*/

NETDEV\_LOG\_OPSET\_ADD\_HOLIDAY =564, /\* 添加假日 \*/

NETDEV\_LOG\_OPSET\_DEL\_HOLIDAY =565, /\* 删除假日 \*/

NETDEV\_LOG\_OPSET\_MOD\_HOLIDAY =566, /\* 修改假日 \*/

NETDEV\_LOG\_OPSET\_ONOFF\_HOLIDAY =567, /\* 开启/关闭假日 \*/

NETDEV\_LOG\_OPSET\_ALLOCATE\_SPACE =568, /\* 容量配置 \*/

NETDEV\_LOG\_OPSET\_HDD\_FULL\_POLICY\_CFG =569, /\* 满策略配置 \*/

NETDEV\_LOG\_OPSET\_AUDIO\_STREAM\_CFG =570, /\* 音频流配置 \*/

NETDEV\_LOG\_OPSET\_ARRAY\_PROPERTY\_CFG =571, /\* 阵列属性配置 \*/

NETDEV\_LOG\_OPSET\_HOT\_SPACE\_DISK\_CFG =572, /\* 热备盘配置 \*/

NETDEV\_LOG\_OPSET\_CREAT\_ARRAY =573, /\* 手动创建阵列 \*/

NETDEV\_LOG\_OPSET\_ONE\_CLICK\_CREAT\_ARRAY =574, /\* 一键创建阵列 \*/

NETDEV\_LOG\_OPSET\_REBUILD\_ARRAY =575, /\* 重建阵列 \*/

NETDEV\_LOG\_OPSET\_DEL\_ARRAY =576, /\* 删除阵列 \*/

NETDEV\_LOG\_OPSET\_ENABLE\_RAID =577, /\* 开启RAID模式 \*/

NETDEV\_LOG\_OPSET\_DISABLE\_RAID =578, /\* 关闭RAID模式 \*/

NETDEV\_LOG\_OPSET\_TEST\_SMART =579, /\* S.M.A.R.T检测 \*/

NETDEV\_LOG\_OPSET\_SMART\_CFG =580, /\* S.M.A.R.T配置 \*/

NETDEV\_LOG\_OPSET\_BAD\_SECTOR\_DETECT =581, /\* 坏道检测 \*/

NETDEV\_LOG\_OPSET\_AUDIO\_ALARM\_DURATION =582, /\* 配置声音报警时长 \*/

NETDEV\_LOG\_OPSET\_CLR\_AUDIO\_ALARM =583, /\* 清除声音报警 \*/

NETDEV\_LOG\_OPSET\_IPC\_TIME\_SYNC\_CFG =584, /\* 配置同步摄像机时间 \*/

NETDEV\_LOG\_OPSET\_ENABLE\_DISK\_GROUP =585, /\* 开启盘组 \*/

NETDEV\_LOG\_OPSET\_DISABLE\_DISK\_GRRUOP =586, /\* 关闭盘组 \*/

NETDEV\_LOG\_OPSET\_ONVIF\_AUTH\_CFG =587, /\* ONVIF认证配置 \*/

NETDEV\_LOG\_OPSET\_8021X\_CFG =588, /\* 配置802.1X \*/

NETDEV\_LOG\_OPSET\_ARP\_PROTECTION\_CFG =589, /\* 配置ARP防攻击 \*/

NETDEV\_LOG\_OPSET\_SMART\_BASIC\_INFO\_CFG =590, /\* 智能报警基本信息配置 \*/

NETDEV\_LOG\_OPSET\_CROSS\_LINE\_DETECT\_CFG =591, /\* 越界检测配置 \*/

NETDEV\_LOG\_OPSET\_INSTRUSION\_DETECT\_CFG =592, /\* 区域入侵配置 \*/

NETDEV\_LOG\_OPSET\_PEOPLE\_COUNT\_CFG =593, /\* 客流量配置 \*/

NETDEV\_LOG\_OPSET\_FACE\_DETECT\_CFG =594, /\* 人脸检测配置 \*/

NETDEV\_LOG\_OPSET\_FISHEYE\_CFG =595, /\* 鱼眼配置 \*/

NETDEV\_LOG\_OPSET\_CUSTOM\_PROTOCOL\_CFG =596, /\* 自定义协议配置 \*/

NETDEV\_LOG\_OPSET\_BEHAVIOR\_SEARCH =597, /\* 行为检索 \*/

NETDEV\_LOG\_OPSET\_FACE\_SEARCH =598, /\* 人脸检索 \*/

NETDEV\_LOG\_OPSET\_PEOPLE\_COUNT =599, /\* 客流量统计 \*/

/\* 维护类 \*/

NETDEV\_LOG\_OPSET\_START\_DVR =600, /\* 开机 \*/

NETDEV\_LOG\_OPSET\_STOP\_DVR =601, /\* 关机 \*/

NETDEV\_LOG\_OPSET\_REBOOT\_DVR =602, /\* 重启设备 \*/

NETDEV\_LOG\_OPSET\_UPGRADE =603, /\* 版本升级 \*/

NETDEV\_LOG\_OPSET\_LOGFILE\_EXPORT =604, /\* 导出日志文件 \*/

NETDEV\_LOG\_OPSET\_CFGFILE\_EXPORT =605, /\* 导出配置文件 \*/

NETDEV\_LOG\_OPSET\_CFGFILE\_IMPORT =606, /\* 导入配置文件 \*/

NETDEV\_LOG\_OPSET\_CONF\_SIMPLE\_INIT =607, /\* 简单恢复配置 \*/

NETDEV\_LOG\_OPSET\_CONF\_ALL\_INIT =608, /\* 恢复出厂配置 \*/

NETDEV\_LOG\_OPSET\_VCA\_BACKUP =700, /\* 智能备份 \*/

NETDEV\_LOG\_OPSET\_3G4G\_CFG =701, /\* 3G/4G配置 \*/

NETDEV\_LOG\_OPSET\_MOUNT\_EXTENDED\_DISK =702, /\* 加载扩展硬盘 \*/

NETDEV\_LOG\_OPSET\_UNMOUNT\_EXTENDED\_DISK =703, /\* 卸载扩展硬盘 \*/

NETDEV\_LOG\_OPSET\_FORCE\_USER\_OFFLINE =704, /\* 强制用户下线 \*/

NETDEV\_LOG\_OPSET\_AUTO\_FUNCTION =709, /\* 自动维护 \*/

NETDEV\_LOG\_OPSET\_IPC\_UPRAGDE =710, /\* 摄像机升级 \*/

NETDEV\_LOG\_OPSET\_RESTORE\_IPC\_DEFAULTS =711, /\* 摄像机恢复默认配置 \*/

NETDEV\_LOG\_OPSET\_ADD\_TRANSACTION =712, /\* 添加交易配置 \*/

NETDEV\_LOG\_OPSET\_MOD\_TRANSACTION =713, /\* 修改交易配置 \*/

NETDEV\_LOG\_OPSET\_DEL\_TRANSACTION =714, /\* 删除交易配置 \*/

NETDEV\_LOG\_OPSET\_POS\_OSD =715, /\* POS显示配置设置 \*/

NETDEV\_LOG\_OPSET\_ADD\_HOT\_SPACE\_DEV =716, /\* 添加备机 \*/

NETDEV\_LOG\_OPSET\_DEL\_HOT\_SPACE\_DEV =717, /\* 删除备机 \*/

NETDEV\_LOG\_OPSET\_MOD\_HOT\_SPACE\_DEV =718, /\* 修改备机 \*/

NETDEV\_LOG\_OPSET\_DEL\_WORK\_DEV =719, /\* 删除工作机 \*/

NETDEV\_LOG\_OPSET\_WORKMODE\_TO\_NORMAL\_CFG =720, /\* 设置工作机模式 \*/

NETDEV\_LOG\_OPSET\_WORKMODE\_TO\_HOTSPACE\_CFG =721, /\* 设置备机模式 \*/

NETDEV\_LOG\_OPSET\_AUTO\_GUARD\_CFG =723, /\* 守望配置 \*/

NETDEV\_LOG\_OPSET\_MULTICAST\_CFG =724, /\* 组播配置 \*/

NETDEV\_LOG\_OPSET\_DEFOCUS\_DETECT\_CFG =725, /\* 虚焦检测配置 \*/

NETDEV\_LOG\_OPSET\_SCENECHANGE\_CFG =726, /\* 场景变更配置 \*/

NETDEV\_LOG\_OPSET\_AUTO\_TRCAK\_CFG =727, /\* 智能跟踪配置 \*/

NETDEV\_LOG\_OPSET\_SORT\_CAMERA\_CFG =728, /\* 通道排序 \*/

NETDEV\_LOG\_OPSET\_WATER\_MARK\_CFG =729, /\* 视频水印配置 \*/

/\* VMS操作日志子类型 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_ALL =1000, /\* 全部子类型 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USER\_LOGIN =1001, /\* 用户登陆 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USER\_LOGOUT =1002, /\* 用户登出 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USER\_START\_OPERATION =1003, /\* 用户启动 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USER\_STOP\_OPERATION =1004, /\* 用户停止 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_NEW\_CONFIG =1005, /\* 新增配置 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_EDIT\_CONFIG =1006, /\* 修改配置 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_DELETE\_CONFIG =1007, /\* 删除配置 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_ENABLE\_CONFIG =1008, /\* 启用配置 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_DISABLE\_CONFIG =1009, /\* 停用配置 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_TEST\_CONFIG =1010, /\* 测试配置 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_OPEN\_DOOR =1011, /\* 开门 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_CLOSE\_DOOR =1012, /\* 关门 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_SARM =1013, /\* 布防 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_DISARM =1014, /\* 撤防 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_BYPASS =1015, /\* 旁路 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_UNBYPASS =1016, /\* 非旁路 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_SHARE\_DEVICE =1017, /\* 云端设备共享 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_UN\_SHARE\_DEVICE =1018, /\* 取消共享 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_START\_CAP\_PAC\_TASK =1019, /\* 开始抓包任务 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_STOP\_CAP\_PAC\_TASK =1020, /\* 停止抓包任务 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_DELETE\_CAPPAC\_TASK=1021, /\* 删除抓包任务 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USB\_INSERT =1022, /\* USB插入 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USB\_PULL\_OUT =1023, /\* USB拔出 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_USB\_FORMAT =1024, /\* USB格式化 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_SYNC\_DEV\_CHL\_INFO =1025, /\* 同步设备通道信息操作 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_PLAYBACK\_LOCK =1026, /\* 锁定录像 \*/

NETDEV\_LOG\_VMS\_OPRERATE\_SUB\_PLAYBACK\_UNLOCK =1027 /\* 解锁录像 \*/

}NETDEV\_LOG\_SUB\_TYPE\_E;

### 告警信息查询类型枚举

typedef enum tagNETDEVFindAlarmTypeEn

{

NETDEV\_FIND\_ALARM\_DISK\_STORAGE\_IS\_FULL = 1, /\* 存储满 \*/

NETDEV\_FIND\_ALARM\_DISK\_STORAGE\_WILL\_FULL = 2, /\* 存储即将满 \*/

NETDEV\_FIND\_ALARM\_STOR\_ERR = 3, /\* 存储错误 \*/

NETDEV\_FIND\_ALARM\_STOR\_ERR\_RECOVER = 4, /\* 存储错误恢复 \*/

NETDEV\_FIND\_ALARM\_MOVE\_DETECT = 5, /\* 运动检测告警 \*/

NETDEV\_FIND\_ALARM\_MOVE\_DETECT\_RECOVER = 6, /\* 运动检测告警恢复 \*/

NETDEV\_FIND\_ALARM\_VIDEO\_TAMPER\_DETECT = 7, /\* 遮挡侦测告警 \*/

NETDEV\_FIND\_ALARM\_VIDEO\_TAMPER\_RECOVER = 8, /\* 遮挡侦测告警恢复 \*/

NETDEV\_FIND\_ALARM\_INPUT\_SWITCH = 9, /\* 输入开关量告警 \*/

NETDEV\_FIND\_ALARM\_INPUT\_SWITCH\_RECOVER = 10, /\* 输入开关量告警恢复\*/

NETDEV\_FIND\_ALARM\_BAND\_CHANGE = 11, /\* 设备出口带宽变更 \*/

NETDEV\_FIND\_ALARM\_IMAGE\_BLURRY = 12, /\* 图像虚焦 \*/

NETDEV\_FIND\_ALARM\_IMAGE\_BLURRY\_RECOVER = 13, /\* 阵列损坏 \*/

NETDEV\_FIND\_ALARM\_SCENE\_CHANGE = 14, /\* 阵列衰退 \*/

NETDEV\_FIND\_ALARM\_ILLEGAL\_ACCESS = 15, /\* 阵列恢复正常 \*/

NETDEV\_FIND\_ALARM\_REPORT\_DEV\_ONLINE = 16, /\* 设备上线 \*/

NETDEV\_FIND\_ALARM\_REPORT\_DEV\_OFFLINE = 17, /\* 设备下线 \*/

NETDEV\_FIND\_ALARM\_REPORT\_DEV\_VIDEO\_LOSS = 18, /\* 视频丢失 \*/

NETDEV\_FIND\_ALARM\_REPORT\_DEV\_VIDEO\_LOSS\_RECOVER = 19, /\* 视频丢失恢复 \*/

NETDEV\_FIND\_ALARM\_COUNT\_PEOPLE = 20, /\* 人数统计 \*/

NETDEV\_FIND\_ALARM\_HEAT\_MAP = 21, /\* 热度图 \*/

NETDEV\_FIND\_ALARM\_LINE\_DETECTOR = 22, /\* 伴线告警 \*/

NETDEV\_FIND\_ALARM\_CELL\_MOTION = 23, /\* 智能运动检测 \*/

NETDEV\_FIND\_ALARM\_LINE\_CROSS = 24, /\* 越界告警 \*/

NETDEV\_FIND\_ALARM\_OBJECTS\_INSIDE = 25, /\* 区域入侵 \*/

NETDEV\_FIND\_ALARM\_ACCESSZONE = 26, /\* 进入区域告警 \*/

NETDEV\_FIND\_ALARM\_LEAVE\_ZONE = 27, /\* 离开区域告警 \*/

NETDEV\_FIND\_ALARM\_HOVER\_ZONE = 28, /\* 区域徘徊告警 \*/

NETDEV\_FIND\_ALARM\_OVER\_ZONE = 29, /\* 翻越围栏告警 \*/

NETDEV\_FIND\_ALARM\_CARE\_ARTICLE = 30, /\* 物品看护告警 \*/

NETDEV\_FIND\_ALARM\_REMAIN\_ARTICLE = 31, /\* 物品遗留告警 \*/

NETDEV\_FIND\_ALARM\_FACE\_DETECTOR = 32, /\* 人脸检测 \*/

NETDEV\_FIND\_ALARM\_GATHER = 33, /\* 人员聚集告警 \*/

NETDEV\_FIND\_ALARM\_GATHER\_RECOVER = 34, /\* 人员聚集告警恢复 \*/

NETDEV\_FIND\_ALARM\_FAST\_MOVE = 35, /\* 快速移动 \*/

NETDEV\_FIND\_ALARM\_ILLEGAL\_PARKED = 36, /\* 违法停车告警 \*/

NETDEV\_FIND\_ALARM\_HUMAN\_SHAPE\_ON = 37, /\* 人形检测告警 \*/

NETDEV\_FIND\_ALARM\_HUMAN\_SHAPE\_OFF = 38, /\* 人形检测告警恢复 \*/

NETDEV\_FIND\_ALARM\_BODY\_TEMPERATURE = 39, /\* 体温异常告警 \*/

NETDEV\_FIND\_ALARM\_NO\_MASK = 40, /\* 未戴口罩告警 \*/

NETDEV\_FIND\_ALARM\_METAL\_DETECT\_ON = 41, /\* 金属检测告警开始 \*/

NETDEV\_FIND\_ALARM\_METAL\_DETECT\_OFF = 42, /\* 金属检测告警结束 \*/

NETDEV\_FIND\_ALARM\_CROWD\_DENSITY\_MINOR\_ON = 43, /\* 人员密度普通告警 \*/

NETDEV\_FIND\_ALARM\_CROWD\_DENSITY\_MINOR\_OFF = 44, /\* 人员密度普通告警恢复 \*/

NETDEV\_FIND\_ALARM\_CROWD\_DENSITY\_MAJOR\_ON = 45, /\* 人员密度中度告警 \*/

NETDEV\_FIND\_ALARM\_CROWD\_DENSITY\_MAJOR\_OFF = 46, /\* 人员密度中度告警恢复 \*/

NETDEV\_FIND\_ALARM\_CROWD\_DENSITY\_CRITICAL\_ON = 47, /\* 人员密度严重告警 \*/

NETDEV\_FIND\_ALARM\_CROWD\_DENSITY\_CRITICAL\_OFF = 48, /\* 人员密度严重告警恢复 \*/

NETDEV\_FIND\_ALARM\_INVALID = 0xFFFF /\* 无效值 \*/

}NETDEV\_FIND\_ALARM\_TYPE\_E;

### 告警日志查询条件枚举

typedef enum tagNETDEVQueryCondType

{

NETDEV\_QUERYCOND\_USERNAME = 0, /\* 用户名称 \*/

NETDEV\_QUERYCOND\_ORGNAME = 1, /\* 组织名称 \*/

NETDEV\_QUERYCOND\_DEVNAME = 2, /\* 设备名称 \*/

NETDEV\_QUERYCOND\_CHNNAME = 3, /\* 通道名称 \*/

NETDEV\_QUERYCOND\_TIME = 4, /\* 时间 \*/

NETDEV\_QUERYCOND\_BUSINESSTYPE = 5, /\* 业务类型 \*/

NETDEV\_QUERYCOND\_OPERATETYPE = 6, /\* 操作类型 \*/

NETDEV\_QUERYCOND\_OPEROBJECT = 7, /\* 操作对象 \*/

NETDEV\_QUERYCOND\_ALARMTYPE = 8, /\* 告警类型 参见枚举[NETDEV\_ALARM\_TYPE\_E](#_告警类型枚举_1)\*/

NETDEV\_QUERYCOND\_ALARMSRCNAME= 9, /\* 告警源名称 \*/

NETDEV\_QUERYCOND\_ALARMLEVEL = 10, /\* 告警级别 \*/

NETDEV\_QUERYCOND\_ALARMCHECKED = 11, /\* 告警是否确认 \*/

NETDEV\_QUERYCOND\_ALARMCHECKUSER = 12, /\* 告警确认用户 \*/

NETDEV\_QUERYCOND\_ALARMCHECKTIME = 13, /\* 告警确认时间 \*/

NETDEV\_QUERYCOND\_ALARM\_DEVID = 14, /\* 告警设备ID \*/

NETDEV\_QUERYCOND\_ALARM\_CHNID = 15, /\* 告警通道ID \*/

NETDEV\_QUERYCOND\_ALARM\_SUBTYPE = 16, /\* 告警子类型 \*/

NETDEV\_QUERYCOND\_ALARM\_SERVER = 17, /\* 归属服务器 \*/

NETDEV\_QUERYCOND\_DOOR\_NUM = 18, /\* 门编号 \*/

NETDEV\_QUERYCOND\_CARD\_NUM = 19, /\* 卡号 \*/

NETDEV\_QUERYCOND\_ALARM\_GENDER = 20, /\* 性别 \*/

NETDEV\_QUERYCOND\_ALARM\_BIRTHDAY = 21, /\* 出生年月 \*/

NETDEV\_QUERYCOND\_MONITOY\_REASON = 22, /\* 布控原因 \*/

NETDEV\_QUERYCOND\_PLATE\_NUM = 23, /\* 车牌号码 \*/

NETDEV\_QUERYCOND\_VEHICLE\_TYPE = 24, /\* 车辆类型 \*/

NETDEV\_QUERYCOND\_PLATE\_COLOR = 25, /\* 车牌颜色 \*/

NETDEV\_QUERYCOND\_VEHICLE\_COLOR = 26, /\* 车身颜色 \*/

NETDEV\_QUERYCOND\_PERSON\_NUMBER = 27, /\* 员工编号\*/

NETDEV\_QUERYCOND\_PERSON\_TYPE = 28, /\* 人员类型\*/

NETDEV\_QUERYCOND\_DIRECT = 29, /\* 方向\*/

NETDEV\_QUERYCOND\_ORGID = 30, /\* 组织ID \*/

NETDEV\_QUERYCOND\_ORGPID = 31, /\* 组织PID \*/

NETDEV\_QUERYCOND\_DEVICEID = 32, /\* 设备ID \*/

NETDEV\_QUERYCOND\_DEVICE\_TYPE = 33, /\* 设备类型 \*/

NETDEV\_QUERYCOND\_DEVICE\_SUBTYPE = 34, /\* 设备子类型 \*/

NETDEV\_QUERYCOND\_CHANNELID = 35, /\* 通道ID \*/

NETDEV\_QUERYCOND\_CHANNEL\_TYPE = 36, /\* 通道类型 \*/

NETDEV\_QUERYCOND\_ONLINE\_STATE = 37, /\* 在线状态 \*/

NETDEV\_DATABASE\_ID = 38, /\* 库ID \*/

NETDEV\_QUERY\_TYPE\_PLATECLASS = 39, /\* 车牌类型 \*/

NETDEV\_QUERYCOND\_RANGE = 40, /\* 告警查询范围 0是设备，1是服务器\*/

NETDEV\_QUERYCOND\_BEGIN\_TIME = 41, /\* 访客预约开始时间\*/

NETDEV\_QUERYCOND\_END\_TIME = 42, /\* 访客预约结束时间\*/

NETDEV\_QUERYCOND\_INTERVIEWEE\_NAME = 43, /\* 受访人姓名\*/

NETDEV\_QUERYCOND\_INTERVIEWEE\_STATUS= 44, /\* 受访人状态\*/

NETDEV\_QUERYCOND\_PARK\_NAME = 45, /\* 停车场名称 \*/

NETDEV\_QUERYCOND\_CONFIDENCE\_LEVEL = 46, /\* 置信度 \*/

NETDEV\_QUERYCOND\_PARK\_TIME = 47, /\* 停车时长 \*/

NETDEV\_QUERYCOND\_CONTRACT\_RULE = 48, /\* 包期规则 \*/

NETDEV\_QUERYCOND\_PAYMENT\_METHOD = 49, /\* 付款方式 \*/

NETDEV\_QUERYCOND\_PASSING\_DIRECTION= 50, /\* 过车方向 \*/

NETDEV\_QUERYCOND\_VEHICLE\_ATTR = 51, /\* 车辆属性 \*/

NETDEV\_QUERYCOND\_STATISTICS\_UNITS = 52, /\* 统计单位 \*/

NETDEV\_QUERYCOND\_EXITENTRANCE\_NAME= 53, /\* 出入口名称 \*/

NETDEV\_QUERYCOND\_PICTURE\_DATA = 54, /\* 图片数据 \*/

NETDEV\_QUERYCOND\_PERSON\_NAME = 55, /\* 人员姓名 \*/

NETDEV\_QUERYCOND\_SIMILARITY = 56, /\* 相似度 \*/

NETDEV\_QUERYCOND\_SEARCH\_TYPE = 57, /\* 检索类型，详见[NETDEV\_SEARCH\_TYPE\_E](#_人脸查找类型) \*/

NETDEV\_QUERYCOND\_ID\_NUMBER = 58, /\* 证件号 \*/

NETDEV\_QUERYCOND\_AGERANGE = 59, /\* 年龄段 \*/

NETDEV\_QUERYCOND\_GLASSES\_STYLE = 61, /\* 眼镜款式 \*/

NETDEV\_QUERYCOND\_SLEEVES\_LENGTH = 62, /\* 上衣长短款式 \*/

NETDEV\_QUERYCOND\_COAT\_COLOR = 63, /\* 上衣颜色 \*/

NETDEV\_QUERYCOND\_TROUSERS\_STYLE = 64, /\* 下衣长短款式 \*/

NETDEV\_QUERYCOND\_TROUSERS\_COLOR = 65, /\* 下衣颜色 \*/

NETDEV\_QUERYCOND\_SNAP\_TOWARD = 66, /\* 抓拍朝向 \*/

NETDEV\_QUERYCOND\_SHOES\_TUBE\_LENGTH= 67, /\* 鞋子长短款式 \*/

NETDEV\_QUERYCOND\_HAIR\_LENGTH = 68, /\* 发型长短 \*/

NETDEV\_QUERYCOND\_BAG\_FLAG = 69, /\* 是否携包 \*/

NETDEV\_QUERYCOND\_SPEED\_TYPE = 70, /\* 速度类型 \*/

NETDEV\_QUERYCOND\_NON\_VEH\_TYPE = 71, /\* 非机动车类型 \*/

NETDEV\_QUERYCOND\_VEH\_BRAND = 72, /\* 车辆品牌 \*/

NETDEV\_QUERYCOND\_VEH\_DATA\_TYPE = 73, /\* 车辆数据类型(0：普通抓拍，1：结构化抓拍) \*/

NETDEV\_QUERYCOND\_PROTOCOL\_TYPE = 74, /\* 设备接入协议 \*/

NETDEV\_QUERYCOND\_RELEVANT\_ROOM = 75, /\* 关联房间 \*/

NETDEV\_QUERYCOND\_LOCK\_SIGNAL = 76, /\* 智能锁信号 \*/

NETDEV\_QUERYCOND\_BIND\_RELATION\_DOORLOCK = 77, /\* 门锁绑定关系 \*/

NETDEV\_QUERYCOND\_BIND\_RELATION\_PERSON\_CARD = 78, /\* 人卡绑定关系 \*/

NETDEV\_QUERYCOND\_PERSONID = 79, /\* 人员ID \*/

NETDEV\_QUERYCOND\_PARKINGLOTID = 80, /\* 停车场ID \*/

NETDEV\_QUERYCOND\_ENTREXITID = 81, /\* 出入口ID \*/

NETDEV\_QUERYCOND\_RECORDID = 82, /\* 记录ID \*/

NETDEV\_QUERYCOND\_VEH\_GROUPINGID = 83, /\* 车辆分组ID \*/

NETDEV\_QUERYCOND\_KEYWORD = 84, /\* 关键字 \*/

NETDEV\_QUERYCOND\_PUBULISH\_TYPE = 85, /\* 发布类型 (0：即时发布,1：排程发布) \*/

NETDEV\_QUERYCOND\_PROGRAM\_NAME = 86, /\* 节目名称 \*/

NETDEV\_QUERYCOND\_ORG\_TYPE = 87, /\* 组织类型，详见[NETDEV\_ORG\_TYPE\_E](#_组织类型枚举) \*/

NETDEV\_QUERYCOND\_VALID\_DATE = 88, /\* 有效日期 \*/

NETDEV\_QUERYCOND\_VALID\_TIME = 89, /\* 有效时间 \*/

NETDEV\_QUERYCOND\_VALID\_WEEK = 90, /\* 有效星期 \*/

NETDEV\_QUERYCOND\_MODEL\_STATUS\_TYPE= 91,

/\* 人脸建模状态, 参见[NETDEV\_MODEL\_STATUS\_E](#_建模状态枚举) \*/

NETDEV\_QUERYCOND\_MASK\_FLAG = 92, /\* 是否戴口罩，详见[NETDEV\_MASK\_FLAG\_E](#_是否戴口罩枚举) \*/

NETDEV\_QUERYCOND\_TEMPERATURE = 93, /\* 人员体温 \*/

NETDEV\_QUERYCOND\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_QUERYCOND\_TYPE\_E;

### 告警类型枚举

typedef enum tagNETDEVAlarmTypeEn

{

NETDEV\_ALARM\_MOVE\_DETECT = 1, /\* 运动检测告警 \*/

NETDEV\_ALARM\_MOVE\_DETECT\_RECOVER = 2, /\* 运动检测告警恢复 \*/

NETDEV\_ALARM\_VIDEO\_LOST = 3, /\* 视频丢失告警 \*/

NETDEV\_ALARM\_VIDEO\_LOST\_RECOVER = 4, /\* 视频丢失告警恢复 \*/

NETDEV\_ALARM\_VIDEO\_TAMPER\_DETECT = 5, /\* 遮挡侦测告警 \*/

NETDEV\_ALARM\_VIDEO\_TAMPER\_RECOVER = 6, /\* 遮挡侦测告警恢复 \*/

NETDEV\_ALARM\_INPUT\_SWITCH = 7, /\* 输入开关量告警 \*/

NETDEV\_ALARM\_INPUT\_SWITCH\_RECOVER = 8, /\* 输入开关量告警恢复 \*/

NETDEV\_ALARM\_TEMPERATURE\_HIGH = 9, /\* 高温告警 \*/

NETDEV\_ALARM\_TEMPERATURE\_LOW = 10, /\* 低温告警 \*/

NETDEV\_ALARM\_TEMPERATURE\_RECOVER = 11, /\* 温度告警恢复 \*/

NETDEV\_ALARM\_AUDIO\_DETECT = 12, /\* 音频异常检测告警 \*/

NETDEV\_ALARM\_AUDIO\_DETECT\_RECOVER = 13, /\* 音频异常检测告警恢复 \*/

NETDEV\_ALARM\_SERVER\_FAULT = 18, /\* 服务器故障 \*/

NETDEV\_ALARM\_SERVER\_NORMAL = 19, /\* 服务器故障恢复 \*/

NETDEV\_ALARM\_REPORT\_DEV\_ONLINE = 201, /\* 设备上线告警 \*/

NETDEV\_ALARM\_REPORT\_DEV\_OFFLINE = 202, /\* 设备下线告警 \*/

NETDEV\_ALARM\_REPORT\_DEV\_REBOOT = 203, /\* 设备重启 \*/

NETDEV\_ALARM\_REPORT\_DEV\_SERVICE\_REBOOT = 204, /\* 服务重启 \*/

NETDEV\_ALARM\_REPORT\_DEV\_CHL\_ONLINE = 205, /\* 视频通道: 上线 \*/

NETDEV\_ALARM\_REPORT\_DEV\_CHL\_OFFLINE = 206, /\* 视频通道: 下线 \*/

NETDEV\_ALARM\_REPORT\_DEV\_DELETE\_CHL = 207, /\* 视频通道: 删除 \*/

NETDEV\_ALARM\_DEVICE\_HIGHTEMP = 246, /\* 异常类：设备高温 \*/

NETDEV\_ALARM\_DEVICE\_LOWTEMP = 247, /\* 异常类：设备低温 \*/

NETDEV\_ALARM\_FAN\_FAULT = 248, /\* 异常类：风扇故障 \*/

NETDEV\_ALARM\_LEDBOX\_HIGHTEMP = 249, /\* 异常类：电箱高温 \*/

NETDEV\_ALARM\_LEDBOX\_SMOKE = 250, /\* 异常类：电箱烟雾告警 \*/

NETDEV\_ALARM\_DEVICE\_HIGHTEMP\_RECOVER = 251, /\* 异常类:设备温度恢复 \*/

NETDEV\_ALARM\_DEVICE\_LOWTEMP\_RECOVER = 252, /\* 异常类:设备温度恢复 \*/

NETDEV\_ALARM\_FAN\_FAULT\_RECOVER = 253, /\* 异常类:风扇故障恢复 \*/

NETDEV\_ALARM\_LEDBOX\_HIGHTEMP\_RECOVER = 254, /\* 异常类:电箱高温恢复 \*/

NETDEV\_ALARM\_LEDBOX\_SMOKE\_RECOVER = 255, /\* 异常类:电箱烟雾告警恢复 \*/

NETDEV\_ALARM\_THERM\_HIGH\_TEMPERATURE = 256, /\* 温度过高报警 \*/

NETDEV\_ALARM\_THERM\_LOW\_TEMPERATURE = 257, /\* 温度过低报警 \*/

NETDEV\_ALARM\_THERM\_ABNORMAL\_TEMPERATURE= 258, /\* 温度异常报警 \*/

/\* 播放器上报告警模块 \*/

NETDEV\_ALARM\_NET\_FAILED = 401, /\* 会话网络错误 \*/

NETDEV\_ALARM\_NET\_TIMEOUT = 402, /\* 会话网络超时 \*/

NETDEV\_ALARM\_SHAKE\_FAILED = 403, /\* 会话交互错误 \*/

NETDEV\_ALARM\_STREAMNUM\_FULL = 404, /\* 流数已经满 \*/

NETDEV\_ALARM\_STREAM\_THIRDSTOP = 405, /\* 第三方停止流 \*/

NETDEV\_ALARM\_FILE\_END = 406, /\* 文件结束 \*/

NETDEV\_ALARM\_RTMP\_CONNECT\_FAIL = 407, /\* RTMP连接失败 \*/

NETDEV\_ALARM\_RTMP\_INIT\_FAIL = 408, /\* RTMP初始化失败\*/

NETDEV\_ALARM\_STREAM\_DOWNLOAD\_OVER = 409, /\* 一体机国标流下载完成 \*/

NETDEV\_ALARM\_PLAYBACK\_FINISH = 410, /\* 回放结束 \*/

NETDEV\_ALARM\_VIDEO\_RECORD\_PART = 411, /\* 录像分段 \*/

NETDEV\_ALARM\_FISHEYE\_STREAM\_EXIST = 412, /\* 鱼眼流存在,仅用于上报 \*/

NETDEV\_ALARM\_FISHEYE\_STREAM\_NOT\_EXIST = 413, /\* 鱼眼流不存在,仅用于上报 \*/

NETDEV\_ALARM\_PTZ\_RESOUCE\_FAIL = 414, /\* 四目全景ptz资源错误 \*/

NETDEV\_ALARM\_PTZ\_STREAM\_EXIST = 415, /\* 四目全景ptz流存在，仅用于上报 \*/

NETDEV\_ALARM\_STREAM\_NOT\_EXIST = 416, /\* 四目全景ptz流不存在，仅用于上报 \*/

NETDEV\_ALARM\_INNER\_TIMEOUT = 417, /\* 内部处理超时 \*/

NETDEV\_ALARM\_STREAM\_NOT\_READY = 418, /\* 流未就绪 \*/

NETDEV\_ALARM\_KEEP\_ALIVE\_FAILED = 419, /\* 保活失败 \*/

NETDEV\_ALARM\_OVER\_ABILITY = 420, /\* 回放能力不足 \*/

NETDEV\_ALARM\_UNAUTHORIZED = 421, /\* 未通过认证 \*/

NETDEV\_ALARM\_FORIBIDDEN = 422, /\* 禁止 \*/

NETDEV\_ALARM\_METHOD\_NOT\_ALLOWED = 423, /\* 不允许该方法 \*/

NETDEV\_ALARM\_PRECONDITION\_FAILED = 424, /\* 预处理失败 \*/

NETDEV\_ALARM\_SESSION\_NOT\_FOUND = 425, /\* 找不到会话 \*/

NETDEV\_ALARM\_NOT\_ENOUGH\_BANDWIDTH2 = 426, /\* 带宽不足(RTSP) \*/

NETDEV\_ALARM\_REALPLAY\_ESTABLISHED = 427, /\* 实况业务已经建立 \*/

NETDEV\_ALARM\_REALPLAY\_RES\_BUSY = 428, /\* 实况业务显示资源忙 \*/

NETDEV\_ALARM\_MULTICAST\_DISABLED = 429, /\* 组播使能关闭 \*/

NETDEV\_ALARM\_MULTICAST\_PORT\_OCCUPIED = 430, /\* 组播端口已被占用 \*/

NETDEV\_ALARM\_MULTICAST\_PORT\_EXHAUSTED = 431, /\* 组播端口已耗尽 \*/

NETDEV\_ALARM\_MULTICAST\_USER\_NOT\_EXIST = 432, /\* 组播用户不存在 \*/

NETDEV\_ALARM\_CHANNEL\_NOT\_ONLINE = 433 /\* 通道不在线 \*/

NETDEV\_ALARM\_TALKBACK\_ENCODED\_INVALID = 434, /\* 语音对讲资源编码无效 \*/

NETDEV\_ALARM\_VOICE\_RES\_USED\_BY\_TALKBACK = 435, /\* 语音资源已被对讲使用 \*/

NETDEV\_ALARM\_TALKBACK\_EXISTS = 436, /\* 语音对讲已存在 \*/

NETDEV\_ALARM\_VOICE\_WORK\_NOT\_EXIST = 437, /\* 语音业务不存在 \*/

NETDEV\_ALARM\_TALKBACK\_TIMEOUT = 438, /\* 建立语音对讲业务超时 \*/

NETDEV\_ALARM\_TALKBACK\_ERROR = 439, /\* 语音对讲失败 \*/

NETDEV\_ALARM\_INNER\_ERROR = 440, /\* 设备侧内部处理错误 \*/

NETDEV\_ALARM\_BAD\_REQUEST = 441, /\* 错误的请求 \*/

NETDEV\_ALARM\_PAYMENT\_REQUIRED = 442, /\* 需要付费 \*/

NETDEV\_ALARM\_NOT\_FOUND = 443, /\* 没有找到 \*/

NETDEV\_ALARM\_NOT\_ACCEPTABLE = 444, /\* 不接受 \*/

NETDEV\_ALARM\_PROXY\_REQUIRED = 445, /\* 代理需要认证 \*/

NETDEV\_ALARM\_REQUEST\_TIMEOUT = 446, /\* 请求超时 \*/

NETDEV\_ALARM\_GONE = 447, /\* 不在服务器 \*/

NETDEV\_ALARM\_LENGTH\_REQUIRED = 448, /\* 需要长度 \*/

NETDEV\_ALARM\_ENTITY\_TOO\_LARGE = 449, /\* 请求实体过长 \*/

NETDEV\_ALARM\_URI\_TOO\_LARGE = 450, /\* 请求-URI过长 \*/

NETDEV\_ALARM\_UNSUPPORTED\_TYPE = 451, /\* 媒体类型不支持 \*/

NETDEV\_ALARM\_NOT\_UNDERSTOOD = 452, /\* 不理解此参数 \*/

NETDEV\_ALARM\_CONFERENCE\_NOT\_FOUND = 453, /\* 找不到会议 \*/

NETDEV\_ALARM\_NOT\_ENOUGH\_BANDWIDTH = 454, /\* 带宽不足(RTSP) \*/

NETDEV\_ALARM\_METHOD\_NOT\_VALID = 455, /\* 此状态下此方法无效 \*/

NETDEV\_ALARM\_HEADER\_NOT\_VALID = 456, /\* 此头部域对该资源无效 \*/

NETDEV\_ALARM\_INVALID\_RANGE = 457, /\* 无效范围 \*/

NETDEV\_ALARM\_PARAMETER\_READ\_ONLY = 458, /\* 参数是只读的 \*/

NETDEV\_ALARM\_AO\_NOT\_ALLOWED = 459, /\* 不允许合控制 \*/

NETDEV\_ALARM\_ONLY\_AO\_ALLOWED = 460, /\* 只允许合控制 \*/

NETDEV\_ALARM\_UNSUPPORTED\_TRANSPORT = 461, /\* 传输方式不支持 \*/

NETDEV\_ALARM\_DESTINATION\_UNREACHABLE = 462, /\* 无法到达目的地址 \*/

NETDEV\_ALARM\_INTERNAL\_SERVER\_ERROR = 463, /\* 服务器内部错误 \*/

NETDEV\_ALARM\_NOT\_IMPLEMENTED = 464, /\* 未实现 \*/

NETDEV\_ALARM\_BAD\_GATEWAY = 465, /\* 网关错误 \*/

NETDEV\_ALARM\_SERVICE\_UNAVAILABLE = 466, /\* 无法得到服务 \*/

NETDEV\_ALARM\_GATEWAY\_TIMEOUT = 467, /\* 网关超时 \*/

NETDEV\_ALARM\_VERSION\_NOT\_SUPPORTED = 468, /\* 不支持此RTSP版本 \*/

NETDEV\_ALARM\_OPTION\_NOT\_SUPPORTED = 469, /\* 不支持选项 \*/

NETDEV\_ALARM\_SESSION\_NOT\_EXIST = 470, /\* 会话不存在 \*/

NETDEV\_ALARM\_UNDEFINED\_ERROR = 471, /\* 未定义错误 \*/

/\* 告警模块 \*/

NETDEV\_ALARM\_DISK\_ERROR = 601, /\* 设备磁盘错误 \*/

NETDEV\_ALARM\_SYS\_DISK\_ERROR = 602, /\* 系统磁盘错误 \*/

NETDEV\_ALARM\_DISK\_ONLINE = 603, /\* 设备磁盘上线 \*/

NETDEV\_ALARM\_SYS\_DISK\_ONLINE = 604, /\* 系统磁盘上线 \*/

NETDEV\_ALARM\_DISK\_OFFLINE = 605, /\* 设备磁盘离线 \*/

NETDEV\_ALARM\_SYS\_DISK\_OFFLINE = 606, /\* 系统磁盘离线 \*/

NETDEV\_ALARM\_DISK\_ABNORMAL = 607, /\* 磁盘异常 \*/

NETDEV\_ALARM\_DISK\_ABNORMAL\_RECOVER = 608, /\* 磁盘异常恢复 \*/

NETDEV\_ALARM\_DISK\_STORAGE\_WILL\_FULL = 609, /\* 磁盘存储空间即将满 \*/

NETDEV\_ALARM\_DISK\_STORAGE\_WILL\_FULL\_RECOVER = 610, /\* 磁盘存储空间即将满恢复 \*/

NETDEV\_ALARM\_DISK\_STORAGE\_IS\_FULL = 611, /\* 设备存储空间满 \*/

NETDEV\_ALARM\_SYS\_DISK\_STORAGE\_IS\_FULL = 612, /\* 系统存储空间满 \*/

NETDEV\_ALARM\_DISK\_STORAGE\_IS\_FULL\_RECOVER= 613, /\* 存储空间满恢复 \*/

NETDEV\_ALARM\_DISK\_RAID\_DISABLED\_RECOVER = 614, /\* 阵列损坏恢复 \*/

NETDEV\_ALARM\_DISK\_RAID\_DEGRADED = 615, /\* 设备阵列衰退 \*/

NETDEV\_ALARM\_SYS\_DISK\_RAID\_DEGRADED = 616, /\* 系统阵列衰退 \*/

NETDEV\_ALARM\_DISK\_RAID\_DISABLED = 617, /\* 设备阵列损坏 \*/

NETDEV\_ALARM\_SYS\_DISK\_RAID\_DISABLED = 618, /\* 系统阵列损坏 \*/

NETDEV\_ALARM\_DISK\_RAID\_DEGRADED\_RECOVER = 619, /\* 阵列衰退恢复 \*/

NETDEV\_ALARM\_STOR\_GO\_FULL = 620, /\* 设备存储即将满告警 \*/

NETDEV\_ALARM\_SYS\_STOR\_GO\_FULL = 621, /\* 系统存储即将满告警 \*/

NETDEV\_ALARM\_ARRAY\_NORMAL = 622, /\* 设备阵列正常 \*/

NETDEV\_ALARM\_SYS\_ARRAY\_NORMAL = 623, /\* 系统阵列正常 \*/

NETDEV\_ALARM\_DISK\_RAID\_RECOVERED = 624, /\* 阵列恢复正常 \*/

NETDEV\_ALARM\_STOR\_ERR = 625, /\* 设备存储错误 \*/

NETDEV\_ALARM\_SYS\_STOR\_ERR = 626, /\* 系统存储错误 \*/

NETDEV\_ALARM\_STOR\_ERR\_RECOVER = 627, /\* 存储错误恢复 \*/

NETDEV\_ALARM\_STOR\_DISOBEY\_PLAN = 628, /\* 未按计划存储 \*/

NETDEV\_ALARM\_STOR\_DISOBEY\_PLAN\_RECOVER = 629, /\* 未按计划存储恢复 \*/

NETDEV\_ALARM\_BANDWITH\_CHANGE = 801, /\* 设备出口带宽变更 \*/

NETDEV\_ALARM\_VIDEOENCODER\_CHANGE = 802, /\* 设备码流配置变更告警 \*/

NETDEV\_ALARM\_IP\_CONFLICT = 803, /\* IP冲突异常告警 \*/

NETDEV\_ALARM\_IP\_CONFLICT\_CLEARED = 804, /\* IP冲突异常告警恢复 \*/

NETDEV\_ALARM\_NET\_OFF = 805, /\* 网络断开异常告警 \*/

NETDEV\_ALARM\_NET\_RESUME\_ON = 806, /\* 网络断开恢复告警 \*/

NETDEV\_ALRAM\_CONFLAG\_DETECT = 920, /\* 火点告警 \*/

NETDEV\_ALARM\_NO\_MASK = 921, /\* 未戴口罩 \*/

NETDEV\_ALARM\_BODY\_TEMPERATURE = 922, /\* 人体温度 \*/

NETDEV\_ALARM\_AREA\_PEOPLE\_COUNT = 923, /\* 区域人数统计告警 \*/

NETDEV\_ALARM\_AREA\_PEOPLE\_COUNT\_RECOVER = 924, /\* 区域人数统计告警恢复 \*/

NETDEV\_ALARM\_ILLEGAL\_ACCESS = 1001, /\* 设备非法访问 \*/

NETDEV\_ALARM\_SYS\_ILLEGAL\_ACCESS = 1002, /\* 系统非法访问 \*/

NETDEV\_ALARM\_LINE\_CROSS = 1003, /\* 越界告警 \*/

NETDEV\_ALARM\_OBJECTS\_INSIDE = 1004, /\* 区域入侵 \*/

NETDEV\_ALARM\_FACE\_RECOGNIZE = 1005, /\* 人脸识别 \*/

NETDEV\_ALARM\_IMAGE\_BLURRY = 1006, /\* 图像虚焦 \*/

NETDEV\_ALARM\_SCENE\_CHANGE = 1007, /\* 场景变更 \*/

NETDEV\_ALARM\_SMART\_TRACK = 1008, /\* 智能跟踪 \*/

NETDEV\_ALARM\_LOITERING\_DETECTOR = 1009, /\* 徘徊检测 \*/

NETDEV\_ALARM\_BANDWIDTH\_CHANGE = 1010, /\* 带宽变更 \*/

NETDEV\_ALARM\_ALLTIME\_FLAG\_END = 1011, /\* 无布防告警结束标记 \*/

NETDEV\_ALARM\_MEDIA\_CONFIG\_CHANGE = 1012, /\* 编码参数变更 \*/

NETDEV\_ALARM\_REMAIN\_ARTICLE = 1013, /\*物品遗留告警 \*/

NETDEV\_ALARM\_PEOPLE\_GATHER = 1014, /\* 人员聚集告警 \*/

NETDEV\_ALARM\_ENTER\_AREA = 1015, /\* 进入区域 \*/

NETDEV\_ALARM\_LEAVE\_AREA = 1016, /\* 离开区域 \*/

NETDEV\_ALARM\_ARTICLE\_MOVE = 1017, /\* 物品搬移 \*/

NETDEV\_ALARM\_SMART\_FACE\_MATCH\_LIST = 1018, /\* 人脸识别黑名单报警 \*/

NETDEV\_ALARM\_SMART\_FACE\_MATCH\_LIST\_RECOVER = 1019, /\* 人脸识别黑名单报警恢复 \*/

NETDEV\_ALARM\_SMART\_FACE\_MISMATCH\_LIST = 1020, /\* 人脸识别不匹配报警 \*/

NETDEV\_ALARM\_SMART\_FACE\_MISMATCH\_LIST\_RECOVER = 1021, /\* 人脸识别不匹配报警恢复 \*/

NETDEV\_ALARM\_SMART\_VEHICLE\_MATCH\_LIST = 1022, /\* 车辆识别匹配报警 \*/

NETDEV\_ALARM\_SMART\_VEHICLE\_MATCH\_LIST\_RECOVER = 1023, /\* 车辆识别匹配报警恢复 \*/

NETDEV\_ALARM\_SMART\_VEHICLE\_MISMATCH\_LIST = 1024, /\* 车辆识别不匹配报警 \*/

NETDEV\_ALARM\_SMART\_VEHICLE\_MISMATCH\_LIST\_RECOVER= 1025, /\* 车辆识别不匹配报警恢复 \*/

NETDEV\_ALARM\_IMAGE\_BLURRY\_RECOVER = 1026, /\* 图像虚焦告警恢复 \*/

NETDEV\_ALARM\_SMART\_TRACK\_RECOVER = 1027, /\* 智能跟踪告警恢复 \*/

NETDEV\_ALARM\_SMART\_READ\_ERROR\_RATE = 1028, /\* 底层数据读取错误率 \*/

NETDEV\_ALARM\_SMART\_SPIN\_UP\_TIME = 1029, /\* 主轴起旋时间 \*/

NETDEV\_ALARM\_SMART\_START\_STOP\_COUNT = 1030, /\* 启停计数 \*/

NETDEV\_ALARM\_SMART\_REALLOCATED\_SECTOR\_COUNT = 1031, /\* 重映射扇区计数 \*/

NETDEV\_ALARM\_SMART\_SEEK\_ERROR\_RATE = 1032, /\* 寻道错误率 \*/

NETDEV\_ALARM\_SMART\_POWER\_ON\_HOURS = 1033, /\* 通电时间累计，出厂后通电的总时间，一般磁盘寿命三万小时 \*/

NETDEV\_ALARM\_SMART\_SPIN\_RETRY\_COUNT = 1034, /\* 主轴起旋重试次数 \*/

NETDEV\_ALARM\_SMART\_CALIBRATION\_RETRY\_COUNT = 1035, /\* 磁头校准重试计数 \*/

NETDEV\_ALARM\_SMART\_POWER\_CYCLE\_COUNT = 1036, /\* 通电周期计数 \*/

NETDEV\_ALARM\_SMART\_POWEROFF\_RETRACT\_COUNT = 1037, /\* 断电返回计数 \*/

NETDEV\_ALARM\_SMART\_LOAD\_CYCLE\_COUNT = 1038, /\* 磁头加载计数 \*/

NETDEV\_ALARM\_SMART\_TEMPERATURE\_CELSIUS = 1039, /\* 温度 \*/

NETDEV\_ALARM\_SMART\_REALLOCATED\_EVENT\_COUNT = 1040, /\* 重映射事件计数 \*/

NETDEV\_ALARM\_SMART\_CURRENT\_PENDING\_SECTOR = 1041, /\* 当前待映射扇区计数 \*/

NETDEV\_ALARM\_SMART\_OFFLINE\_UNCORRECTABLE = 1042, /\* 脱机无法校正的扇区计数 \*/

NETDEV\_ALARM\_SMART\_UDMA\_CRC\_ERROR\_COUNT = 1043, /\* 奇偶校验错误率 \*/

NETDEV\_ALARM\_SMART\_MULTI\_ZONE\_ERROR\_RATE = 1044, /\* 多区域错误率 \*/

NETDEV\_ALARM\_RESOLUTION\_CHANGE = 1045, /\* 分辨率变更 \*/

NETDEV\_ALARM\_MANUAL = 1401, /\* 手动告警 \*/

NETDEV\_ALARM\_ALARMHOST\_COMMON = 1402, /\* 报警点事件 \*/

NETDEV\_ALARM\_DOORHOST\_COMMON = 1403, /\* 门禁事件 \*/

NETDEV\_ALARM\_FACE\_NOT\_MATCH = 1411, /\* 人脸对比失败 \*/

NETDEV\_ALARM\_FACE\_MATCH\_SUCCEED = 1412, /\* 人脸对比成功 \*/

NETDEV\_ALARM\_VIDEO\_CHL\_REACH\_UPPER\_LIMIT = 1413, /\* 编码通道数量已达到上限 \*/

NETDEV\_ALARM\_VEHICLE\_BLACK\_LIST = 1420, /\* 车辆识别黑名单报警 \*/

NETDEV\_ALARM\_HUMAN\_SHAPE\_DETECTION = 1421, /\* 人形检测 \*/

NETDEV\_ALARM\_HUMAN\_SHAPE\_DETECTION\_RECOVER = 1422, /\* 人形检测告警恢复 \*/

NETDEV\_ALARM\_NOT\_WORN\_SAFETYHELMET = 1423, /\* 未佩戴安全帽报警 \*/

NETDEV\_ALARM\_NOT\_WORN\_WORKCLOTHES = 1424, /\* 未穿戴工作服报警 \*/

NETDEV\_ALARM\_FAST\_MOVING = 1425, /\* 快速移动报警 \*/

NETDEV\_ALARM\_NOT\_WORN\_CHEFHAT = 1426, /\* 未带厨师帽报警 \*/

NETDEV\_ALARM\_TELEPHONING = 1427, /\* 打电话告警 \*/

NETDEV\_ALARM\_SMOKING = 1428, /\* 吸烟告警 \*/

NETDEV\_ALARM\_CROWD\_DENSITY\_MINOR = 1429, /\* 人员密度普通告警 \*/

NETDEV\_ALARM\_CROWD\_DENSITY\_MINOR\_CLEARED = 1430, /\* 人员密度普通告警恢复 \*/

NETDEV\_ALARM\_CROWD\_DENSITY\_MAJOR = 1431, /\* 人员密度中度告警 \*/

NETDEV\_ALARM\_CROWD\_DENSITY\_MAJOR\_CLEARED = 1432, /\* 人员密度中度告警恢复 \*/

NETDEV\_ALARM\_CROWD\_DENSITY\_CRITICAL = 1433, /\* 人员密度严重告警 \*/

NETDEV\_ALARM\_CROWD\_DENSITY\_CRITICAL\_CLEARED = 1434, /\* 人员密度严重告警恢复 \*/

NETDEV\_ALARM\_ACCESS\_ELEVATOR = 1435, /\* 入梯报警 \*/

NETDEV\_ALARM\_ACCESS\_ELEVATOR\_CLEARED = 1436, /\* 入梯报警恢复 \*/

NETDEV\_ALARM\_INVALID = 0xFFFF /\* 无效值 s\*/

}NETDEV\_ALARM\_TYPE\_E;

### 人脸查找类型枚举

typedef enum tagNETDEVSearchType

{

NETDEV\_SEARCH\_TYPE\_FACE = 0, /\* 人脸检索 \*/

NETDEV\_SEARCH\_TYPE\_FACE\_COMPARE\_SUCCESS = 1, /\* 人脸比对成功 \*/

NETDEV\_SEARCH\_TYPE\_FACE\_COMPARE\_FAIL = 2, /\* 人脸比对失败 \*/

NETDEV\_SEARCH\_TYPE\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_SEARCH\_TYPE\_E;

### 组织类型枚举

typedef enum tagNETDEVOrgType

{

NETDEV\_ORG\_TYPE\_GENERAL = 0, /\* 普通组织 \*/

NETDEV\_ORG\_TYPE\_CLOUD = 1, /\* 云端组织 \*/

NETDEV\_ORG\_TYPE\_VIRTUAL = 2, /\* 逻辑组织 \*/

NETDEV\_ORG\_TYPE\_FAVORITES = 3, /\* 收藏夹 \*/

NETDEV\_ORG\_TYPE\_DOMAIN = 4, /\* 域名组织 \*/

NETDEV\_ORG\_TYPE\_DOORGROUP = 5, /\* 门组 \*/

NETDEV\_ORG\_TYPE\_DEPT = 6, /\* 部门组织 \*/

NETDEV\_ORG\_TYPE\_APARTMENT = 7, /\* 公寓组织 \*/

NETDEV\_ORG\_TYPE\_INVALID = 0XFF /\* 无效值 \*/

}NETDEV\_ORG\_TYPE\_E;

### 建模状态枚举

typedef enum tagNETDEVModelStatus

{

NETDEV\_MODEL\_STATUS\_TYPE\_UNMODELED = 0, /\* 0:未建模 \*/

NETDEV\_MODEL\_STATUS\_TYPE\_SUCCEED = 1, /\* 1:已建模 \*/

NETDEV\_MODEL\_STATUS\_TYPE\_FAILED = 2, /\* 2:建模失败 \*/

NETDEV\_MODEL\_STATUS\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_MODEL\_STATUS\_E;

### 是否戴口罩枚举

typedef enum tagNETDEVMaskFlag

{

NETDEV\_MASK\_FLAG\_UNKNOW = 0, /\* 未知 \*/

NETDEV\_MASK\_FLAG\_NOT\_WEAR = 1, /\* 不戴 \*/

NETDEV\_MASK\_FLAG\_WEAR = 2, /\* 戴 \*/

NETDEV\_MASK\_FLAG\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_MASK\_FLAG\_E;

### 查询条件逻辑类型枚举

typedef enum tagNETDEVQueryCondLogic

{

NETDEV\_QUERYCOND\_LOGIC\_EQUAL = 0, /\* 查询条件逻辑类型：等于 \*/

NETDEV\_QUERYCOND\_LOGIC\_GREATER = 1, /\* 查询条件逻辑类型：大于 \*/

NETDEV\_QUERYCOND\_LOGIC\_LESS = 2, /\* 查询条件逻辑类型：小于 \*/

NETDEV\_QUERYCOND\_LOGIC\_NO\_LESS = 3, /\* 查询条件逻辑类型：不小于 \*/

NETDEV\_QUERYCOND\_LOGIC\_NO\_GREATER = 4, /\* 查询条件逻辑类型：不大于 \*/

NETDEV\_QUERYCOND\_LOGIC\_NO\_EQUAL = 5, /\* 查询条件逻辑类型：不等于 \*/

NETDEV\_QUERYCOND\_LOGIC\_DIM\_QUERY = 6, /\* 查询条件逻辑类型：模糊查询 \*/

NETDEV\_QUERYCOND\_LOGIC\_CONTAIN = 7, /\* 查询条件逻辑类型：包括 \*/

NETDEV\_QUERYCOND\_LOGIC\_ASC\_ORDER = 8, /\* 查询条件逻辑类型：升序 \*/

NETDEV\_QUERYCOND\_LOGIC\_DESC\_ORDER = 9 /\* 查询条件逻辑类型：降序 \*/

}NETDEV\_QUERYCOND\_LOGICTYPE\_E;

### 告警子类型枚举

typedefenumtagNETDEVAlarmSubType

{

NETDEV\_ALARM\_SUB\_TYPE\_NOT\_CONFIGURED =0, /\*未配置报警子类型\*/

NETDEV\_DEV\_TYPE\_BASE =100000UL, /\*安防报警基数\*/

NETDEV\_DEV\_TROUBLE =100001, /\*设备故障\*/

NETDEV\_DEV\_ALARM\_EMERGENCY =100002, /\*设备紧急报警\*/

/\*报警设备报警类型定义\*/

NETDEV\_ALARM\_EVENT\_BASE =100200UL, /\*报警类型定义起始位置\*/

NETDEV\_ALARM\_EVENT\_FIRE\_ALARM =100201, /\*火警\*/

NETDEV\_ALARM\_EVENT\_KEYPADFIRE =100202, /\*键盘火警报警\*/

NETDEV\_ALARM\_EVENT\_KEYPADEMERGENCY =100203, /\*键盘紧急报警\*/

NETDEV\_ALARM\_EVENT\_KEYPADMEDICAL =100204, /\*键盘求助报警\*/

NETDEV\_ALARM\_EVENT\_DURESSCODEUSED =100205, /\*胁持码操作\*/

NETDEV\_ALARM\_EVENT\_UNAUTHORIZEDENTRY =100206, /\*未授权进入\*/

NETDEV\_ALARM\_EVENT\_BURGLARPOINTALARM =100207, /\*盗警防区报警\*/

NETDEV\_ALARM\_EVENT\_SUPERVISORY =100208, /\*监测故障\*/

NETDEV\_ALARM\_EVENT\_FIRETROUBLE =100209, /\*火警故障\*/

NETDEV\_ALARM\_EVENT\_FIREPOINTTROUBLE =100210, /\*火警防区故障\*/

NETDEV\_ALARM\_EVENT\_BURGLARPOINTTROUBLE =100211, /\*盗警防区故障\*/

NETDEV\_ALARM\_EVENT\_CANCELALARM =100212, /\*消警\*/

NETDEV\_ALARM\_EVENT\_DISARM =100213, /\*撤防操作成功\*/

NETDEV\_ALARM\_EVENT\_ARM =100214, /\*布防操作成功\*/

NETDEV\_ALARM\_EVENT\_FORCEDBYPASS =100215, /\*强制旁路操作\*/

NETDEV\_ALARM\_EVENT\_BYPASS =100216, /\*防区旁路操作\*/

NETDEV\_ALARM\_EVENT\_POINTRESTORAL =100217, /\*防区报警恢复\*/

NETDEV\_ALARM\_EVENT\_FAILTOCLOSE =100218, /\*布防操作失败\*/

NETDEV\_ALARM\_EVENT\_FAILTOOPEN =100219, /\*撤防操作失败\*/

NETDEV\_ALARM\_EVENT\_HISTORYBUFFERDUMP =100220, /\*事件记录满\*/

NETDEV\_ALARM\_EVENT\_TESTMSG =100221, /\*测试用\*/

NETDEV\_ALARM\_EVENT\_ZONETROUBLE =100222, /\*防区故障\*/

NETDEV\_ALARM\_EVENT\_ZONETROUBLERESTORE =100223, /\*防区故障恢复\*/

NETDEV\_ALARM\_EVENT\_ZONESHORT =100224, /\*防区短路\*/

NETDEV\_ALARM\_EVENT\_ZONEOPEN =100225, /\*防区开路\*/

NETDEV\_ALARM\_EVENT\_ZONENORMAL =100226, /\*防区正常\*/

NETDEV\_ALARM\_EVENT\_MXDEVCLOSE =100227, /\*MX设备布防\*/

NETDEV\_ALARM\_EVENT\_MXDEVOPEN =100228, /\*MX设备撤防\*/

NETDEV\_ALARM\_EVENT\_MXDEV1ZONEALARM =100229, /\*MX设备1防区报警\*/

NETDEV\_ALARM\_EVENT\_MXDEV2ZONEALARM =100230, /\*MX设备2防区报警\*/

NETDEV\_ALARM\_EVENT\_MXDEV3ZONEALARM =100231, /\*MX设备3防区报警\*/

NETDEV\_ALARM\_EVENT\_MXDEV1ZONERESTAORE =100232, /\*MX设备1防区恢复\*/

NETDEV\_ALARM\_EVENT\_MXDEV2ZONERESTAORE =100233, /\*MX设备2防区恢复\*/

NETDEV\_ALARM\_EVENT\_MXDEV3ZONERESTAORE =100234, /\*MX设备3防区恢复\*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_AC =100235, /\*交流电源故障\*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_BATTERY =100236, /\*备用电源故障\*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_SNDREPORT =100237, /\*报告发送故障\*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_CONTROL =100238, /\*控制故障\*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_MXBUS =100239, /\*MPX总线故障\*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_RADIOSND =100240, /\*无线接收故障\*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_ASSISTPOWER =100241, /\*辅助电源故障\*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_OPTION =100242, /\*option设备故障\*/

NETDEV\_ALARM\_EVENT\_LINK\_BREAK =100243, /\*联机断开\*/

NETDEV\_ALARM\_EVENT\_LINK\_RESUME =100244, /\*联机正常\*/

NETDEV\_ALARM\_EVENT\_VOLTAGE\_ALARM =100245, /\*直流欠压报警\*/

NETDEV\_ALARM\_EVENT\_EXTERNDEV\_TROUBLE =100246, /\*扩展设备故障报警\*/

NETDEV\_ALARM\_EVENT\_EXTERNDEV\_RESUME =100247, /\*扩展设备恢复报警\*/

NETDEV\_ALARM\_EVENT\_POINT\_DISARM =100248, /\*点位撤防\*/

NETDEV\_ALARM\_EVENT\_POINT\_ARM =100249, /\*点位布防\*/

NETDEV\_ALARM\_EVENT\_FORCE\_OPEN =100250, /\*报警主机被强制打开\*/

NETDEV\_ALARM\_EVENT\_BYPASS\_FAILED =100251, /\*旁路操作失败\*/

NETDEV\_ALARM\_EVENT\_RESET\_FAILED =100252, /\*复位操作失败\*/

NETDEV\_ALARM\_EVENT\_POINT\_FIRETROUBLE =100253, /\*点位故障\*/

NETDEV\_ALARM\_EVENT\_POINT\_SEPARATE =100254, /\*点位隔离\*/

NETDEV\_ALARM\_EVENT\_POINT\_ACTION =100255, /\*点位动作\*/

NETDEV\_ALARM\_EVENT\_POINT\_RESUME =100256, /\*点位恢复\*/

NETDEV\_ALARM\_EVENT\_ROB\_ACTION =100257, /\*劫警\*/

/\*报警事件新增\*/

NETDEV\_ALARM\_EVENT\_MISSED\_ARM =100260, /\*没有布防\*/

NETDEV\_ALARM\_EVENT\_MISSED\_DISARM =100261, /\*没有撤防\*/

NETDEV\_ALARM\_EVENT\_AUXILARY\_RST =100262, /\*辅助报警恢复\*/

NETDEV\_ALARM\_EVENT\_AUXILARY =100263, /\*辅助报警\*/

NETDEV\_ALARM\_EVENT\_BURGLARY\_RST =100264, /\*窃警恢复\*/

NETDEV\_ALARM\_EVENT\_BURGLARY =100265, /\*窃警\*/

NETDEV\_ALARM\_EVENT\_BYPASS\_RST =100266, /\*旁路恢复\*/

NETDEV\_ALARM\_EVENT\_CANCEL =100267, /\*取消\*/

NETDEV\_ALARM\_EVENT\_DUPLICATE =100268, /\*重复\*/

NETDEV\_ALARM\_EVENT\_DURESS\_RST =100269, /\*挟持报警恢复\*/

NETDEV\_ALARM\_EVENT\_DURESS =100270, /\*挟持报警\*/

NETDEV\_ALARM\_EVENT\_FIRE\_RST =100271, /\*火警恢复\*/

NETDEV\_ALARM\_EVENT\_FIRE\_TROUBLE =100272, /\*防火探头故障\*/

NETDEV\_ALARM\_EVENT\_FIRE\_TRB\_RST =100273, /\*防火探头故障恢复\*/

NETDEV\_ALARM\_EVENT\_MED\_RST =100274, /\*医疗救助复位\*/

NETDEV\_ALARM\_EVENT\_PANIC =100275, /\*紧急报警\*/

NETDEV\_ALARM\_EVENT\_PANIC\_RST =100276, /\*紧急报警恢复\*/

NETDEV\_ALARM\_EVENT\_RESET =100277, /\*复位\*/

NETDEV\_ALARM\_EVENT\_RELAY\_SHORTED =100278, /\*继电器短路\*/

NETDEV\_ALARM\_EVENT\_RELAY\_RST =100279, /\*继电器恢复\*/

NETDEV\_ALARM\_EVENT\_RELAY\_TROUBLE =100280, /\*继电器故障\*/

NETDEV\_ALARM\_EVENT\_TAMPER =100281, /\*防拆报警\*/

NETDEV\_ALARM\_EVENT\_TAMPER\_RST =100282, /\*防拆报警恢复\*/

NETDEV\_ALARM\_EVENT\_TROUBLE =100283, /\*故障\*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_RST =100284, /\*故障恢复\*/

/\*乌审旗对接东方电子DF8900电力综合自动化系统新增\*/

NETDEV\_ALARM\_EVENT\_DFE\_REMOTE\_CONTROL=100290, /\*遥控\*/

NETDEV\_ALARM\_EVENT\_DFE\_SYNC =100291, /\*同期\*/

NETDEV\_ALARM\_EVENT\_DFE\_LIFTING =100292, /\*升降\*/

NETDEV\_ALARM\_EVENT\_DFE\_SWITCH\_NORMAL =100293, /\*开关正常变位\*/

NETDEV\_ALARM\_EVENT\_DFE\_SWITCH\_ACCIDENT=100294, /\*开关事故变位\*/

NETDEV\_ALARM\_EVENT\_DFE\_KINFE\_GATE =100295, /\*刀闸正常变位\*/

/\*垦华监狱晨鹰哨位集成箱报警\*/

NETDEV\_ALARM\_EVENT\_ESCAPE =100296, /\*晨鹰哨位事件

NETDEV\_ALARM\_EVENT\_ESCAPE\_END =100297, /\*晨鹰哨位事件

NETDEV\_ALARM\_EVENT\_VIOLENCE =100298, /\*晨鹰哨位事件

NETDEV\_ALARM\_EVENT\_VIOLENCE\_END =100299, /\*晨鹰哨位事件

NETDEV\_ALARM\_EVENT\_ATTACKED =100300, /\*晨鹰哨位事件

NETDEV\_ALARM\_EVENT\_ATTACKED\_END =100301, /\*晨鹰哨位事件

NETDEV\_ALARM\_EVENT\_NATURAL\_DISASTER =100302, /\*晨鹰哨位事件

NETDEV\_ALARM\_EVENT\_NATURAL\_DISASTER\_END =100303, /\*晨鹰哨位事件

NETDEV\_ALARM\_EVENT\_ONE\_KEY\_ALARM =100304, /\*晨鹰哨位事件

NETDEV\_ALARM\_EVENT\_ONE\_KEY\_ALARM\_END =100305, /\*晨鹰哨位事件

/\*潍北监狱中电瑞达高压电网报警\*/

NETDEV\_ALARM\_EVENT\_CERC\_NORMAL =100306, /\*中电瑞达事件

NETDEV\_ALARM\_EVENT\_CERC\_TOUCH =100307, /\*中电瑞达事件

NETDEV\_ALARM\_EVENT\_CERC\_BREAK =100308, /\*中电瑞达事件

NETDEV\_ALARM\_EVENT\_CERC\_SHORT =100309, /\*中电瑞达事件

NETDEV\_ALARM\_EVENT\_CERC\_COMMU\_ERR =100310, /\*中电瑞达事件

,

/\*潍北监狱艾博信相控阵雷达和红外光栅报警\*/ ,

NETDEV\_ALARM\_EVENT\_HYPERSEE\_DDR\_ALARM =100311, /\*艾博信事件

NETDEV\_ALARM\_EVENT\_HYPERSEE\_DDR\_ALARM\_RST =100312, /\*艾博信事件

NETDEV\_ALARM\_EVENT\_HYPERSEE\_IR\_ALARM =100313, /\*艾博信事件

NETDEV\_ALARM\_EVENT\_HYPERSEE\_IR\_ALARM\_RST =100314, /\*艾博信事件

/\*潍北监狱中科融通陀螺仪报警\*/

NETDEV\_ALARM\_EVENT\_UBI\_ALARM =100315, /\*中科融通事件

/\*新疆中泰园区中科华讯美国西南周界震动电缆报警\*/

NETDEV\_ALARM\_EVENT\_HXTECH\_CABLE\_ALARM =100316, /\*华讯周界事件

/\*精华隆数据\*/

NETDEV\_ALARM\_EVENT\_DEVICE\_UNDER\_VOLTAGE\_RST =100317, /\*设备欠压恢复模式\*/

NETDEV\_ALARM\_EVENT\_ZONE\_NOT\_READY =100318, /\*防区未准备\*/

NETDEV\_ALARM\_EVENT\_ZONE\_NOT\_READY\_RST =100319, /\*防区未准备恢复\*/

NETDEV\_ALARM\_EVENT\_TELEPHONE\_CABLE\_FAULT =100320, /\*电话线故障\*/

NETDEV\_ALARM\_EVENT\_TELEPHONE\_CABLE\_FAULT\_RST=100321, /\*电话线故障恢复\*/

NETDEV\_ALARM\_EVENT\_AC\_POWER\_FAULT =100322, /\*交流电故障\*/

/\*一体机对接艾立安告警\*/

NETDEV\_ALARM\_EVENT\_ZONE\_TOUCH =100323, /\*防区触发报警\*/

NETDEV\_ALARM\_EVENT\_ZONE\_TOUCH\_RESTORE =100324, /\*防区触发恢复\*/

NETDEV\_ALARM\_EVENT\_CAPTIVE =100325, /\*被撬\*/

NETDEV\_ALARM\_EVENT\_CAPTIVE\_RST =100326, /\*被撬恢复\*/

NETDEV\_ALARM\_EVENT\_UNDERVOLTAGE =100327, /\*欠压\*/

NETDEV\_ALARM\_EVENT\_UNDERVOLTAGE\_RST =100328, /\*欠压恢复\*/

NETDEV\_ALARM\_EVENT\_TROUBLE\_BATTERY\_RST =100329, /\*备用电池故障恢复\*/

NETDEV\_ALARM\_EVENT\_DEV\_POWER\_REBOOT =100330, /\*设备上电重启\*/

NETDEV\_ALARM\_EVENT\_TEST\_FAILURE =100331, /\*设备测试失败\*/

NETDEV\_ALARM\_EVENT\_OPERATOR\_TOUCH\_LONG =100332, /\*长按键操作\*/

NETDEV\_ALARM\_EVENT\_REMOTE\_ARM =100333, /\*遥控布防\*/

NETDEV\_ALARM\_EVENT\_REMOTE\_DISARM =100334, /\*遥控撤防\*/

NETDEV\_ALARM\_EVENT\_REMOTE\_STAY\_ARM =100335, /\*遥控留守布防\*/

NETDEV\_ALARM\_EVENT\_PWD\_ARM =100336, /\*密码布防设备\*/

NETDEV\_ALARM\_EVENT\_PWD\_DISARM =100337, /\*密码撤防设备\*/

NETDEV\_ALARM\_EVENT\_PWD\_STAY\_DEV =100338, /\*密码留守布防设备\*/

NETDEV\_ALARM\_EVENT\_PWD\_ARM\_PART =100339, /\*密码布防分区\*/

NETDEV\_ALARM\_EVENT\_PWD\_DISARM\_PART =100340, /\*密码撤防分区\*/

NETDEV\_ALARM\_EVENT\_PWD\_STAY\_PART =100341, /\*密码留守布防分区\*/

NETDEV\_ALARM\_EVENT\_PWD\_ARM\_ZONE =100342, /\*密码布防防区\*/

NETDEV\_ALARM\_EVENT\_PWD\_DISARM\_ZONE =100343, /\*密码撤防防区\*/

NETDEV\_ALARM\_EVENT\_USER\_ARM =100344, /\*用户布防设备\*/

NETDEV\_ALARM\_EVENT\_USER\_DISARM =100345, /\*用户撤防设备\*/

NETDEV\_ALARM\_EVENT\_USER\_STAY\_DEV =100346, /\*用户留守布防设备\*/

NETDEV\_ALARM\_EVENT\_USER\_CLEAN\_ALARM =100347, /\*用户清除报警显示\*/

NETDEV\_ALARM\_EVENT\_PWD\_OPEN\_DOOR =100348, /\*密码开门\*/

NETDEV\_ALARM\_EVENT\_KEY\_OPEN\_DOOR =100349, /\*钥匙开门\*/

NETDEV\_ALARM\_EVENT\_REMOTE\_OPEN\_DOOR =100350, /\*遥控开门\*/

NETDEV\_ALARM\_EVENT\_SWIPE\_ARM =100351, /\*刷卡布防\*/

NETDEV\_ALARM\_EVENT\_SWIPE\_DISARM =100352, /\*刷卡撤防\*/

NETDEV\_ALARM\_EVENT\_SWIPE\_LOCK =100353, /\*刷卡闭锁\*/

NETDEV\_ALARM\_EVENT\_SWIPE\_UNLOCK =100354, /\*刷卡开锁\*/

NETDEV\_ALARM\_EVENT\_SWIPE\_RECORD =100355, /\*刷卡记录\*/

NETDEV\_ALARM\_EVENT\_SWIPE\_ATTENDANCE\_IN=100356, /\*刷卡考勤入\*/

NETDEV\_ALARM\_EVENT\_SWIPE\_ATTENDANCE\_OUT=100357, /\*刷卡考勤出\*/

NETDEV\_ALARM\_EVENT\_SWIPE\_TURNON\_LIGHT =100358, /\*刷卡开灯\*/

NETDEV\_ALARM\_EVENT\_SWIPE\_OFF\_LIGHT =100359, /\*刷卡关灯\*/

NETDEV\_ALARM\_EVENT\_SWIPE\_CARD\_NUM =100360, /\*刷卡上报卡物理编号\*/

NETDEV\_ALARM\_EVENT\_MANUAL\_OPEN =100361, /\*手动开门/开灯\*/

NETDEV\_ALARM\_EVENT\_ZONE\_OPEN\_DOOR =100362, /\*防区触发开门\*/

NETDEV\_ALARM\_EVENT\_ZONE\_DOOR\_LONGTIME=100363, /\*防区长时间未触发布防\*/

NETDEV\_ALARM\_EVENT\_ZONE\_ARM =100364, /\*防区触发布防\*/

NETDEV\_ALARM\_EVENT\_ZONE\_DISARM =100365, /\*防区触发撤防\*/

NETDEV\_ALARM\_EVENT\_TIMER\_ARM =100366, /\*定时布防\*/

NETDEV\_ALARM\_EVENT\_TIMER\_DISARM =100367, /\*定时撤防\*/

NETDEV\_ALARM\_EVENT\_CENTRAL\_COMM\_NORMAL =100368, /\*中心通讯正常\*/

NETDEV\_ALARM\_EVENT\_CENTRAL\_HANDSHAKE\_FAILED =100369, /\*中心握手失败\*/

NETDEV\_ALARM\_EVENT\_CENTRAL\_RESPONSE\_FAILED =100370, /\*中心应答失败\*/

NETDEV\_ALARM\_EVENT\_USER\_CONFIRMATION =100371, /\*用户接警确认\*/

NETDEV\_ALARM\_EVENT\_USER\_TIMEOUT =100372, /\*用户接警超时\*/

NETDEV\_ALARM\_EVENT\_TELEP\_RINGING =100373, /\*电话模块接收到振铃信号\*/

NETDEV\_ALARM\_EVENT\_TELEP\_ARM =100374, /\*电话模块接收到远程撤布防一位操作密码\*/

NETDEV\_ALARM\_EVENT\_TELEP\_STATUS\_OFFLINE =100375, /\*电话线掉线\*/

NETDEV\_ALARM\_EVENT\_TELEP\_STATUS\_ONLINE =100376, /\*电话线在线\*/

NETDEV\_ALARM\_EVENT\_MSG\_SEND\_SUCC =100377, /\*短信发送成功\*/

NETDEV\_ALARM\_EVENT\_MSG\_SEND\_FAILED =100378, /\*短信发送失败\*/

NETDEV\_ALARM\_EVENT\_SWITCH\_CLOSE =100379, /\*操作开关合上\*/

NETDEV\_ALARM\_EVENT\_SWITCH\_OFF =100380, /\*操作开关断开\*/

NETDEV\_ALARM\_EVENT\_TEMPER\_CHANGE =100381, /\*温度变化上报\*/

NETDEV\_ALARM\_EVENT\_HUMIDITY\_CHANGE =100382, /\*湿度变化上报\*/

NETDEV\_ALARM\_EVENT\_DEVID\_WRONG =100383, /\*设备ID错误\*/

NETDEV\_ALARM\_EVENT\_ELEC\_HIGH\_ARM =100384, /\*单个电子围栏高压布防\*/

NETDEV\_ALARM\_EVENT\_ELEC\_LOW\_ARM =100385, /\*单个电子围栏低压布防\*/

NETDEV\_ALARM\_EVENT\_ELEC\_ALARM\_RST =100386, /\*单个电子围栏报警复位\*/

NETDEV\_ALARM\_EVENT\_ELEC\_DISARM =100387, /\*单个电子围栏撤防\*/

NETDEV\_ALARM\_EVENT\_ELEC\_ARM =100388, /\*单个电子围栏电压值布防\*/

NETDEV\_ALARM\_EVENT\_KEY\_ALARM\_OVER =100389, /\*键盘所有报警处理完成\*/

NETDEV\_ALARM\_EVENT\_KEY\_PROL\_TYPE =100390, /\*键盘上报协议类型（主要针对电子围栏）\*/

NETDEV\_ALARM\_EVENT\_MOD\_DEV\_ADDR =100391, /\*键盘修改设备地址（主要针对电子围栏）\*/

NETDEV\_ALARM\_EVENT\_FORWARD\_CMD =100392, /\*通讯设备转发中心命令\*/

NETDEV\_ALARM\_EVENT\_BASE\_VALUE =110000UL, /\*报警类型定义起始位置\*/

NETDEV\_ALARM\_EVENT\_IMMEDIATELY\_DEFEND\_ALARM =110001, /\*即时防区报警\*/

NETDEV\_ALARM\_EVENT\_IMMEDIATELY\_DEFEND\_ALARM\_RST=110002, /\*即时防区报警恢复\*/

NETDEV\_ALARM\_EVENT\_ALL\_DAY\_NO\_VOICE\_ALARM =110003, /\*24小时无声防区报警\*/

NETDEV\_ALARM\_EVENT\_ALL\_DAY\_NO\_VOICE\_ALARM\_RST=110004, /\*24小时无声防区报警恢复\*/

NETDEV\_ALARM\_EVENT\_ALL\_DAY\_VOICE\_ALARM =110005, /\*24小时有声防区报警\*/

NETDEV\_ALARM\_EVENT\_ALL\_DAY\_VOICE\_ALARM\_RST =110006, /\*24小时有声防区报警恢复\*/

NETDEV\_ALARM\_EVENT\_BOARD\_DEFEND\_ALARM =110007, /\*周界防区报警\*/

NETDEV\_ALARM\_EVENT\_BOARD\_DEFEND\_ALARM\_RST =110008, /\*周界防区报警恢复\*/

NETDEV\_ALARM\_EVENT\_INNER\_DELAY\_DEFEND\_ALARM =110009, /\*内部延时防区报警\*/

NETDEV\_ALARM\_EVENT\_INNER\_DELAY\_DEFEND\_ALARM\_RST=110010, /\*内部延时防区报警恢复\*/

NETDEV\_ALARM\_EVENT\_DELAY\_DEFEND\_ALARM =110011, /\*延时防区报警\*/

NETDEV\_ALARM\_EVENT\_DELAY\_DEFEND\_ALARM\_RST =110012, /\*延时防区报警恢复\*/

NETDEV\_ALARM\_EVENT\_EXPAND\_MODEL\_FAULT\_ALARM=110013, /\*拓展模块故障\*/

NETDEV\_ALARM\_EVENT\_EXPAND\_MODEL\_FAULT\_ALARM\_RST=110014, /\*拓展模块故障恢复\*/

NETDEV\_ALARM\_EVENT\_OUTER\_DISARMED =110015, /\*外出撤防\*/

NETDEV\_ALARM\_EVENT\_OUTER\_ARMED =110016, /\*外出布防\*/

NETDEV\_ALARM\_EVENT\_KEY\_DISARMED =110017, /\*钥匙防区撤防\*/

NETDEV\_ALARM\_EVENT\_KEY\_ARMED =110018, /\*钥匙防区布防\*/

NETDEV\_ALARM\_EVENT\_WIRELESS\_NET\_WORK\_EXCEPTION =110019, /\*无线网络异常\*/

NETDEV\_ALARM\_EVENT\_WIRELESS\_NET\_WORK\_EXCEPTION\_RST=110020, /\*无线网络异常恢复\*/

NETDEV\_ALARM\_EVENT\_WIRED\_NET\_WORK\_BREAK =110021, /\*有线网络故障\*/

NETDEV\_ALARM\_EVENT\_WIRED\_NET\_WORK\_BREAK\_RST =110022, /\*有线网络故障恢复\*/

NETDEV\_ALARM\_EVENT\_SOFT\_DEFEND\_URGENCY\_ALARM =110023, /\*软防区紧急报警\*/

NETDEV\_ALARM\_EVENT\_ARMED\_STAY =110024, /\*留守布防\*/

NETDEV\_ALARM\_EVENT\_IMMEDIATELY\_ARMED =110025, /\*即时布防\*/

/\*乌审旗对接昆仑传感（烟感、水浸、继电器报警）报警新增\*/

NETDEV\_ALARM\_EVENT\_SMOKE\_ALARM =110026, /\*昆仑传感烟感报警\*/

NETDEV\_ALARM\_EVENT\_SMOKE\_ALARM\_END =110027, /\*昆仑传感烟感报警停止\*/

NETDEV\_ALARM\_EVENT\_WATER\_ALARM =110028, /\*昆仑传感水浸报警\*/

NETDEV\_ALARM\_EVENT\_WATER\_ALARM\_END =110029, /\*昆仑传感水浸报警停止\*/

NETDEV\_ALARM\_EVENT\_IO\_ON =110030, /\*昆仑传感.开关量开\*/

NETDEV\_ALARM\_EVENT\_IO\_OFF =110031, /\*昆仑传感.开关量关\*/

/\*长飞振动光纤新增报警定义值\*/

NETDEV\_ALARM\_EVENT\_TOUNCH\_ALARM =110032, /\*触碰报警\*/

NETDEV\_ALARM\_EVENT\_INVADE\_ALARM =110033, /\*入侵报警\*/

NETDEV\_ALARM\_EVENT\_BROKEN\_ALARM =110034, /\*断裂报警\*/

NETDEV\_ALARM\_EVENT\_HELP =110036, /\*紧急求助\*/

NETDEV\_ALARM\_EVENT\_DESTROY =110037, /\*破坏报警\*/

NETDEV\_ALARM\_EVENT\_ACS =110038, /\*门禁报警\*/

NETDEV\_ALARM\_EVENT\_OTHERS =110039, /\*其它报警\*/

NETDEV\_ALARM\_EVENT\_ZONE\_ALARM =110040, /\*安标报警模块.防区报警\*/

NETDEV\_ALARM\_EVENT\_ZONE\_ALARM\_RST =110041, /\*安标报警模块.防区恢复\*/

NETDEV\_ALARM\_EVENT\_GATO\_ALARM\_FENCE =110042, /\*围栏报警\*/

NETDEV\_ALARM\_EVENT\_GATO\_ALARM\_HOST =110043, /\*主机报警\*/

NETDEV\_ALARM\_EVENT\_GATO\_ALARM\_ETCZONE =110044, /\*扩展防区报警\*/

NETDEV\_ALARM\_EVENT\_GATO\_ALARM\_OFFLINE =110045, /\*通信掉线\*/

NETDEV\_ALARM\_EVENT\_GATO\_ALARM\_SWITCH1 =110046, /\*防区开关量1报警\*/

NETDEV\_ALARM\_EVENT\_GATO\_ALARM\_SWITCH2 =110047, /\*防区开关量2报警\*/

NETDEV\_ALARM\_EVENT\_STEAL =110048, /\*窃盗报警\*/

NETDEV\_ALARM\_EVENT\_GAS =110049, /\*燃气报警\*/

NETDEV\_ALARM\_EVENT\_SAVE =110050, /\*救护报警\*/

NETDEV\_ALARM\_EVENT\_INVADE\_TROUBLE =110051, /\*入侵加故障报警\*/

NETDEV\_ALARM\_EVENT\_SYS\_TROUBLE =110052, /\*系统故障\*/

NETDEV\_ALARM\_EVENT\_OPTICAL\_FIBER\_TROUBLE =110053, /\*光纤故障\*/

/\*烽火科技报警主机报警子类型\*/

NETDEV\_ALARM\_EVENT\_MOTION\_DETECTION =110062, /\*运动检测报警\*/

NETDEV\_ALARM\_EVENT\_CALL\_REMOVE =110063, /\*呼叫解除报警\*/

NETDEV\_ALARM\_EVENT\_NOISE =110064, /\*噪音报警\*/

NETDEV\_ALARM\_EVENT\_DELAYED =110065, /\*延时报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_INPUT\_ONE =110066, /\*报警输入1报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_INPUT\_TWO =110067, /\*报警输入2报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_CALL\_EMERGENCY =110068, /\*紧急呼叫\*/

NETDEV\_ALARM\_EVENT\_ALARM\_CALL\_CONSULTANT=110069, /\*咨询呼叫\*/

NETDEV\_ALARM\_EVENT\_DOOR\_MAGNET\_OPEN =110070, /\*门磁打开\*/

NETDEV\_ALARM\_EVENT\_DOOR\_MAGNET\_CLOSE =110071, /\*门磁关闭\*/

NETDEV\_ALARM\_EVENT\_DOOR\_LOCK\_OPEN =110072, /\*门锁打开\*/

NETDEV\_ALARM\_EVENT\_DOOR\_LOCK\_CLOSE =110073, /\*门锁关闭\*/

/\*世邦报警柱报警子类型\*/

NETDEV\_ALARM\_EVENT\_ALARM\_PORT\_ONE =110074, /\*端口1报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_PORT\_TWO =110075, /\*端口2报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_PORTEX =110076, /\*扩展端口报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_TUMULT =110077, /\*喧哗报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_110 =110078, /\*110报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_119 =110079, /\*119报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_120 =110080, /\*120报警\*/

/\*蓝星报警主机报警子类型\*/

NETDEV\_ALARM\_EVENT\_ALARM\_FINDERCONNECT\_FAILED =110081, /\*探测器通讯失败\*/

NETDEV\_ALARM\_EVENT\_ALARM\_OUTAGE =110082, /\*断电报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_DISASSEMBLE =110083, /\*拆机报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_CIRCUIT =110084, /\*断路报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_TOUCHNET =110085, /\*触网报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_PREVENTCUT =110086, /\*防剪报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_COCKING =110087, /\*拉紧报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_SLACK =110088, /\*松弛报警\*/

NETDEV\_ALARM\_EVENT\_ALARM\_ZONE\_FORBIDDEN =110089, /\*防区禁用\*/

/\*建大仁科\*/

NETDEV\_ALARM\_EVENT\_HIGH\_TEMP =110090, /\*温度超上限报警\*/

NETDEV\_ALARM\_EVENT\_LOW\_TEMP =110091, /\*温度超下限报警\*/

NETDEV\_ALARM\_EVENT\_HIGH\_HUMI =110092, /\*湿度超上限报警\*/

NETDEV\_ALARM\_EVENT\_LOW\_HUMI =110093, /\*湿度超下限报警\*/

NETDEV\_ALARM\_MONITOR\_DEVICE\_ONLINE =110094, /\*监控点设备在线\*/

NETDEV\_ALARM\_MONITOR\_DEVICE\_OFFLINE =110095, /\*监控点设备离线\*/

/\*世邦报警柱报警子类型--新增子类型\*/

NETDEV\_ALARM\_EVENT\_ALARM\_CALL\_OUT\_RING =110110, /\*呼出振铃\*/

NETDEV\_ALARM\_EVENT\_ALARM\_CALL\_IN =110111, /\*普通呼入\*/

NETDEV\_ALARM\_EVENT\_ALARM\_CALL\_END =110112, /\*通话挂断\*/

NETDEV\_ALARM\_EVENT\_ALARM\_BROADCAST\_TO\_SLAVE\_BEGIN =110113,/\*对主机汉化广播开始\*/

NETDEV\_ALARM\_EVENT\_ALARM\_BROADCAST\_TO\_MAST\_BEGIN =110114,/\*对分机汉化广播开始\*/

NETDEV\_ALARM\_EVENT\_ALARM\_BROADCAST\_END =110115, /\*广播结束\*/

NETDEV\_ALARM\_EVENT\_ALARM\_CALL\_INTERPHONE =110116, /\*对讲接通\*/

/\*门禁上报事件类型定义\*/

NETDEV\_DOOR\_EVENT\_BASE =100400UL, /\*门禁事件类型定义起始位置\*/

NETDEV\_DOOR\_EVENT\_DISCONNECT =100401, /\*通信中断

NETDEV\_DOOR\_EVENT\_INFRARED\_ALARM =100402, /\*红外报警\*/

NETDEV\_DOOR\_EVENT\_ILLICIT\_OPEN\_DOOR =100403, /\*非法开门\*/

NETDEV\_DOOR\_EVENT\_COERCE\_ALARM =100404, /\*胁迫报警\*/

NETDEV\_DOOR\_EVENT\_OPEN\_TIMEOUT =100405, /\*超时未开门\*/

NETDEV\_DOOR\_EVENT\_CLOSE\_TIMEOUT =100406, /\*超时未关门\*/

NETDEV\_DOOR\_EVENT\_ANOMALOUS\_CARD =100407, /\*异常刷卡\*/

NETDEV\_DOOR\_EVENT\_CARD\_OPEN\_DOOR =100408, /\*刷卡开门\*/

NETDEV\_DOOR\_EVENT\_REMOTE\_OPEN\_DOOR =100409, /\*远程开门\*/

NETDEV\_DOOR\_EVENT\_MANUAL\_OPEN\_DOOR =100410, /\*手动开门\*/

NETDEV\_DOOR\_EVENT\_EMERGENCY\_START =100411, /\*紧急事件开始\*/

NETDEV\_DOOR\_EVENT\_EMERGENCY\_STOP =100412, /\*紧急事件结束\*/

NETDEV\_DOOR\_EVENT\_OPEN\_SUCCESS =100413, /\*开门成功\*/

NETDEV\_DOOR\_EVENT\_OPEN\_FAILED =100414, /\*开门失败\*/

NETDEV\_DOOR\_EVENT\_CLOSE =100415, /\*关门事件\*/

NETDEV\_DOOR\_EVENT\_CLOSE\_FAILED =100416, /\*关门失败\*/

NETDEV\_DOOR\_EVENT\_CARD\_NO\_AUTH =100417, /\*卡无权限\*/

NETDEV\_DOOR\_EVENT\_NO\_DATE\_LIMIT =100418, /\*无期限\*/

NETDEV\_DOOR\_EVENT\_CARD\_ILLEGAL =100419, /\*卡非法\*/

NETDEV\_DOOR\_EVENT\_FIREPORTECT\_ALRAM =100420, /\*消防报警\*/

NETDEV\_DOOR\_EVENT\_CANCEL\_FIREPORTECT =100421, /\*取消消防\*/

NETDEV\_DOOR\_EVENT\_PRYING\_RESIST\_ALRAM =100422, /\*防撬报警\*/

NETDEV\_DOOR\_EVENT\_SUPERPASSWD\_OPEN\_DOOR =100423, /\*超级密码开门\*/

NETDEV\_DOOR\_EVENT\_COERCECODE\_IN =100424, /\*胁迫码进入\*/

NETDEV\_DOOR\_EVENT\_DOOR\_STATUS\_OPEN =100425, /\*门打开\*/

NETDEV\_DOOR\_EVENT\_DOOR\_STATUS\_CLOSE =100426, /\*门关闭\*/

NETDEV\_DOOR\_EVENT\_DOOR\_FORCED\_OPEN =100427, /\*门强开\*/

NETDEV\_DOOR\_EVENT\_DOOR\_KEEP\_OPEN =100428, /\*门保持打开\*/

NETDEV\_DOOR\_EVENT\_LOCK\_FAILURE =100429, /\*锁失效\*/

NETDEV\_DOOR\_EVENT\_LOST\_STOLEN\_CARD =100430, /\*丢失或被盗卡\*/

/\*达实门禁对接新增\*/

NETDEV\_DOOR\_EVENT\_PASSWD\_WRONG =100431, /\*密码错误\*/

NETDEV\_DOOR\_EVENT\_ALWAYS\_CLOSED =100432, /\*常闭未通过\*/

NETDEV\_DOOR\_EVENT\_CARD\_PASSWD\_ENTER =100433, /\*卡加密码进入\*/

NETDEV\_DOOR\_EVENT\_CARD\_PASSWD\_WRONG =100434, /\*卡加密码错误\*/

NETDEV\_DOOR\_EVENT\_MORE\_CARD\_OPEN =100435, /\*多卡开门\*/

NETDEV\_DOOR\_EVENT\_FIRST\_CARD\_OPEN =100436, /\*首卡开门\*/

NETDEV\_DOOR\_EVENT\_CARD\_OUT\_DATE =100437, /\*卡过期限\*/

NETDEV\_DOOR\_EVENT\_CARD\_WATCH =100438, /\*巡更卡\*/

/\*海康门禁对接新增\*/

NETDEV\_DOOR\_EVENT\_REMOTE\_CLOSE\_DOOR =100439, /\*远程关门\*/

NETDEV\_DOOR\_EVENT\_REMOTE\_ALWAYS\_OPEN =100440, /\*远程常开\*/

NETDEV\_DOOR\_EVENT\_REMOTE\_ALWAYS\_CLOSE =100441, /\*远程常关\*/

NETDEV\_DOOR\_EVENT\_LEGAL\_CARD\_PASS =100442, /\*合法卡认证通过\*/

NETDEV\_DOOR\_EVENT\_CARD\_AND\_PSW\_PASS =100443, /\*刷卡加密码认证通过\*/

NETDEV\_DOOR\_EVENT\_CARD\_AND\_PSW\_FAIL =100444, /\*刷卡加密码认证失败）\*/

NETDEV\_DOOR\_EVENT\_CARD\_AND\_PSW\_TIMEOUT =100445, /\*数卡加密码认证超时\*/

NETDEV\_DOOR\_EVENT\_CARD\_AND\_PSW\_OVER\_TIME=100446, /\*刷卡加密码超次\*/

NETDEV\_DOOR\_EVENT\_INVALID\_CARD =100447, /\*无此卡号\*/

NETDEV\_DOOR\_EVENT\_DOOR\_BUTTON\_PRESS =100448, /\*开门按钮按下\*/

NETDEV\_DOOR\_EVENT\_DOOR\_BUTTON\_RELEASE =100449, /\*开门按钮松开\*/

NETDEV\_DOOR\_EVENT\_CARD\_READER\_DESMANTLE\_ALARM =100450, /\*读卡器防拆报警\*/

NETDEV\_DOOR\_EVENT\_CARD\_READER\_DESMANTLE\_RESUME =100451, /\*读卡器防拆恢复\*/

NETDEV\_DOOR\_EVENT\_DOOR\_MAGNET\_OPEN =100452, /\*门磁打开（正常开门）\*/

NETDEV\_DOOR\_EVENT\_DOOR\_MAGNET\_CLOSE =100453, /\*门磁关闭（正常关门\*/

NETDEV\_DOOR\_EVENT\_DOOR\_MAGNET\_OPEN\_ABNORMAL=100454, /\*门磁异常打开（门异常打开）\*/

NETDEV\_DOOR\_EVENT\_DOOR\_MAGNET\_OPEN\_TIMEOUT =100455, /\*门磁打开超时

NETDEV\_DOOR\_EVENT\_ANTI\_SNEAK\_FAIL =100456, /\*反潜回认证失败\*/

NETDEV\_DOOR\_EVENT\_INTERLOCK\_DOOR\_NOT\_CLOSE =100457, /\*互锁门未关闭\*/

NETDEV\_DOOR\_EVENT\_MULTI\_VERIFY\_SUCCESS =100458, /\*多重认证成功\*/

NETDEV\_DOOR\_EVENT\_LEADER\_CARD\_OPEN\_BEGIN =100459, /\*首卡开门开始\*/

NETDEV\_DOOR\_EVENT\_LEADER\_CARD\_OPEN\_END =100460, /\*首卡开门结束\*/

NETDEV\_DOOR\_EVENT\_ALWAYS\_OPEN\_BEGIN =100461, /\*常开状态开始\*/

NETDEV\_DOOR\_EVENT\_ALWAYS\_OPEN\_END =100462, /\*常开状态结束\*/

NETDEV\_DOOR\_EVENT\_ALWAYS\_CLOSE\_BEGIN =100463, /\*常关状态开始\*/

NETDEV\_DOOR\_EVENT\_ALWAYS\_CLOSE\_END =100464, /\*常关状态结束\*/

NETDEV\_DOOR\_EVENT\_KEYPRESS\_OPEN\_DOOR =100465, /\*按键开门\*/

/\*纽贝尔门禁新协议对接事件定义\*/

NETDEV\_DOOR\_EVENT\_COMMU\_INTERRUPT =100481, /\*通讯中断\*/

NETDEV\_DOOR\_EVENT\_COMMU\_RESTORE =100482, /\*通讯恢复\*/

NETDEV\_DOOR\_EVENT\_LEGAL\_CREDIT\_CARD =100483, /\*合法刷卡\*/

NETDEV\_DOOR\_EVENT\_LEGAL\_USER\_ID =100484, /\*合法用户ID\*/

NETDEV\_DOOR\_EVENT\_CARD\_NO\_REG =100485, /\*卡未注册\*/

NETDEV\_DOOR\_EVENT\_CARD\_UNAUTH =100486, /\*卡未授权\*/

NETDEV\_DOOR\_EVENT\_CARD\_DISABLED =100487, /\*卡已停用\*/

NETDEV\_DOOR\_EVENT\_CARD\_EXPIRED =100488, /\*卡已过期\*/

NETDEV\_DOOR\_EVENT\_INVALID\_TIME =100489, /\*无效时段\*/

NETDEV\_DOOR\_EVENT\_PSW\_ERROR =100490, /\*密码错误\*/

NETDEV\_DOOR\_EVENT\_PROHIBITED =100491, /\*禁止通行\*/

NETDEV\_DOOR\_EVENT\_REQ\_CENTER\_OPEN\_DOOR =100492, /\*请求中心开门\*/

NETDEV\_DOOR\_EVENT\_LEGAL\_CARD\_OPEN\_DOOR =100493, /\*合法卡开门\*/

NETDEV\_DOOR\_EVENT\_SUPER\_CARD\_OPEN\_DOOR =100494, /\*超级卡开门\*/

NETDEV\_DOOR\_EVENT\_CENTER\_OPEN\_DOOR =100496, /\*中心开门\*/

NETDEV\_DOOR\_EVENT\_LINK\_OPEN\_DOOR =100497, /\*联动开门\*/

NETDEV\_DOOR\_EVENT\_FORCE\_OPEN\_DOOR =100498, /\*强行开门\*/

NETDEV\_DOOR\_EVENT\_DOOR\_IS\_OPEN =100499, /\*门被打开\*/

NETDEV\_DOOR\_EVENT\_DOOR\_IS\_CLOSE =100500, /\*门被关闭\*/

NETDEV\_DOOR\_EVENT\_STRESS\_ALARM =100503, /\*胁迫报警\*/

NETDEV\_DOOR\_EVENT\_TRAILING\_ALARM =100505, /\*尾随报警\*/

NETDEV\_DOOR\_EVENT\_DOOR\_LOCK =100506, /\*门锁死\*/

NETDEV\_DOOR\_EVENT\_LIFT\_DOOR\_LOCK =100507, /\*门解除锁死\*/

NETDEV\_DOOR\_EVENT\_DOOR\_NOT\_CLOSE =100508, /\*门未关好\*/

NETDEV\_DOOR\_EVENT\_NEED\_PSW =100509, /\*需要密码\*/

NETDEV\_DOOR\_EVENT\_NEED\_FINGERPRINT =100510, /\*需要指纹\*/

NETDEV\_DOOR\_EVENT\_REQ\_INTERCOM =100511, /\*请求对讲\*/

NETDEV\_DOOR\_EVENT\_EMERGENCY =100512, /\*紧急事件\*/

NETDEV\_DOOR\_EVENT\_PATROL\_LEGAL\_CREDIT\_CARD =100513, /\*巡更合法刷卡\*/

NETDEV\_DOOR\_EVENT\_PATROL\_LEGAL\_FINGERPRINT =100514, /\*巡更合法指纹\*/

NETDEV\_DOOR\_EVENT\_PATROL\_CARD\_NOT\_REG =100515, /\*巡更卡未注册\*/

NETDEV\_DOOR\_EVENT\_NOT\_PATROL\_CARD =100516, /\*非巡更员刷卡\*/

NETDEV\_DOOR\_EVENT\_NORMAL\_PATROL =100517, /\*正常巡更\*/

NETDEV\_DOOR\_EVENT\_ADVANCE\_PATROL =100518, /\*提前巡更\*/

NETDEV\_DOOR\_EVENT\_TIMEOUT\_PATROL =100519, /\*超时巡更\*/

NETDEV\_DOOR\_EVENT\_INVALID\_PATROL =100520, /\*无效巡更\*/

NETDEV\_DOOR\_EVENT\_NOT\_PATROL =100521, /\*未巡更\*/

NETDEV\_DOOR\_EVENT\_TH\_ALARM =100522, /\*温湿度报警\*/

NETDEV\_DOOR\_EVENT\_TH\_RESTORE =100523, /\*温湿度恢复\*/

NETDEV\_DOOR\_EVENT\_TH\_RECORD =100524, /\*温湿度记录\*/

NETDEV\_DOOR\_EVENT\_SMOKE\_ALARM =100525, /\*烟感报警\*/

NETDEV\_DOOR\_EVENT\_SMOKE\_RESTORE =100526, /\*烟感恢复\*/

NETDEV\_DOOR\_EVENT\_SMOKE\_RECORD =100527, /\*烟感记录\*/

/\*微耕门禁新协议对接事件定义\*/

NETDEV\_DOOR\_EVENT\_PC\_CONTROL =100528, /\*刷卡不通过：电脑控制\*/

NETDEV\_DOOR\_EVENT\_LIMITED\_CARD\_NUM =100529, /\*刷卡不通过：受刷卡次数限制\*/

NETDEV\_DOOR\_EVENT\_REASON\_UNKNOWN =100530, /\*刷卡不通过：原因不明\*/

NETDEV\_DOOR\_EVENT\_CONTROLLER\_ON =100531, /\*控制器上电\*/

NETDEV\_DOOR\_EVENT\_CONTROLLER\_RESERT =100532, /\*控制器复位\*/

NETDEV\_DOOR\_EVENT\_DOOR\_FORCE\_CLOSE =100533, /\*强制关门\*/

NETDEV\_DOOR\_EVENT\_DOOR\_OFFLINE =100534, /\*门不在线\*/

NETDEV\_DOOR\_EVENT\_FIRE =100535, /\*火警\*/

NETDEV\_DOOR\_EVENT\_CALL\_EMERGENCY =100536, /\*呼叫紧急报警\*/

NETDEV\_DOOR\_PERSON\_FACE\_DOOR =100537, /\*人脸刷门\*/

NETDEV\_DOOR\_PERSON\_INVALE\_DOOR =100538, /\*人脸非法刷门\*/

/\*中控门禁对接事件定义\*/

NETDEV\_DOOR\_EVENT\_FINGERPRINT\_OPEN\_DOOR =100553, /\*指纹开门\*/

NETDEV\_DOOR\_EVENT\_ANTI\_SUBMARINE =100554, /\*反潜报警\*/

NETDEV\_DOOR\_EVENT\_PRESS\_MISTAKE =100555, /\*错按报警\*/

NETDEV\_DOOR\_EVENT\_PF\_PW\_RF\_OPEN\_DOOR =100556, /\*指纹、密码或卡验证开门\*/

/\*泰科门禁对接事件定义\*/

NETDEV\_DOOR\_EVENT\_DOOR\_FORCED\_OPEN\_CLEARED =100557, /\*门强制打开恢复\*/

/\*通用门禁对接中控事件定义\*/

NETDEV\_DOOR\_EVENT\_BLACKLIST =100558, /\*黑名单\*/

NETDEV\_DOOR\_EVENT\_FAILED\_SEND\_CMD =100559, /\*发送命令失败\*/

NETDEV\_DOOR\_EVENT\_FAILED\_OPEN\_DOOE\_MU\_CARD =100560, /\*多卡开门失败\*/

NETDEV\_DOOR\_EVENT\_CARD\_INVALID\_PERIOD =100561, /\*非有效时间段刷卡\*/

NETDEV\_DOOR\_EVENT\_CARD\_REPORT\_LOST =100562, /\*卡已挂失\*/

NETDEV\_DOOR\_EVENT\_TAMPER\_ALARM =100563, /\*防拆报警\*/

NETDEV\_DOOR\_EVENT\_SWIPE\_CARD\_WHEN\_NORMAL\_OPEN =100564, /\*常开时间段内刷卡\*/

NETDEV\_DOOR\_EVENT\_OPEN\_DOOR\_BY\_EMERGENCY\_PWD =100565, /\*紧急密码开门\*/

NETDEV\_DOOR\_EVENT\_OPEN\_DOOR\_WHEN\_NORMAL\_OPEN =100566, /\*常开时间段开门\*/

NETDEV\_DOOR\_EVENT\_EXIT\_BUTTON\_DURING\_ILLEGAL\_PERIOD=100567,/\*无效时间段按出门按钮\*/

NETDEV\_DOOR\_EVENT\_DOOR\_ALREADY\_OPEN =100568, /\*门已打开\*/

NETDEV\_DOOR\_EVENT\_DOOR\_ALREADY\_CLOSE =100569, /\*门已关闭\*/

NETDEV\_DOOR\_EVENT\_ILLEGAL\_PERIOD =100570, /\*非法时间段\*/

NETDEV\_DOOR\_EVENT\_OPEN\_DOOR\_BY\_DURESS\_PWD =100571, /\*胁迫密码开门\*/

NETDEV\_DOOR\_EVENT\_NO\_MASK =100572, /\*未戴口罩\*/

NETDEV\_DOOR\_EVENT\_BODY\_TEMPERATURE =100573, /\*体温异常\*/

NETDEV\_DOOR\_EVENT\_M0NITOR\_INPUT\_ACTIVATED = 100574, /\* 监视器处于活动状态 \*/

NETDEV\_DOOR\_EVENT\_M0NITOR\_INPUT\_CLEAR = 100575, /\* 监视器已清除 \*/

NETDEV\_DOOR\_EVENT\_END =100999, /\*门禁事件类型定义结束位置\*/

#define NETDEV\_IS\_DOOR\_EVENT(x) (100400 <= (x) && (x) < NETDEV\_DOOR\_EVENT\_END)

}NETDEV\_ALARM\_SUBTYPE\_E;

### 告警所属类型枚举

typedef enum tagNETDEVAlarmSrcBelongType

{

NETDEV\_ALARM\_SRC\_BELONG\_TYPE\_OTHER = 0, /\* 其他 \*/

NETDEV\_ALARM\_SRC\_BELONG\_TYPE\_CHL = 1, /\* 通道类告警 \*/

NETDEV\_ALARM\_SRC\_BELONG\_TYPE\_DEV = 2, /\* 设备类告警 \*/

NETDEV\_ALARM\_SRC\_BELONG\_TYPE\_INVALID = 0xffff /\* 无效值 \*/

}NETDEV\_ALARM\_SRC\_BELONG\_TYPE\_E;

### 成员性别枚举

typedef enum tagNETDEVGenderType

{

NETDEV\_GENDER\_TYPE\_UNKNOW = 0, /\* 0-未知的性别 \*/

NETDEV\_GENDER\_TYPE\_MAN = 1, /\* 1-男 \*/

NETDEV\_GENDER\_TYPE\_WOMAN = 2, /\* 2-女 \*/

NETDEV\_GENDER\_TYPE\_UNEXPLAINED = 9, /\* 9-未说明的性别 \*/

NETDEV\_GENDER\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_GENDER\_TYPE\_E;

### 证件类型枚举

typedef enum tagNETDEVCertificateType

{

NETDEV\_CERTIFICATE\_TYPE\_ID = 0, /\*0:身份证 \*/

NETDEV\_CERTIFICATE\_TYPE\_IC = 1, /\* 1:IC卡 \*/

NETDEV\_CERTIFICATE\_TYPE\_PASSPORT = 2, /\* 2:护照 \*/

NETDEV\_CERTIFICATE\_TYPE\_DRIVING\_LICENSE = 3, /\* 3:行驶证 \*/

NETDEV\_CERTIFICATE\_TYPE\_OTHER = 99, /\* 99:其他 \*/

NETDEV\_CERTIFICATE\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_ID\_TYPE\_E;

### 文件信息枚举

typedef enum tagNETDEVFileType

{

NETDEV\_TYPE\_FOLDER = 0, /\* 文件夹 \*/

NETDEV\_TYPE\_FILE = 1, /\* 文件 \*/

NETDEV\_TYPE\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_FILE\_TYPE\_E;

### 人脸处理结果状态码枚举

typedef enum tagNETDEVPersonResultCode

{

NETDEV\_PERSON\_CODE\_SUCCEED = 0, /\* 成功 \*/

NETDEV\_PERSON\_CODE\_COMMON\_FAIL = 1, /\* 通用执行失败 \*/

NETDEV\_PERSON\_CODE\_SENDING = 2, /\* 下发中 \*/

NETDEV\_PERSON\_CODE\_DEV\_NOT\_SUPPORT = 4, /\* 设备不支持 \*/

NETDEV\_PERSON\_CODE\_ARGORITHM\_INIT\_FAIL = 1000, /\* 算法初始化失败 \*/

NETDEV\_PERSON\_CODE\_FACE\_DETECT\_FAIL = 1001, /\* 人脸检测失败 \*/

NETDEV\_PERSON\_CODE\_PICTURE\_NO\_FACE = 1002, /\* 图片未检测到人脸 \*/

NETDEV\_PERSON\_CODE\_JPEG\_DECODE\_FAIL = 1003, /\* jpeg照片解码失败 \*/

NETDEV\_PERSON\_CODE\_PICTURE\_QUALITY\_LOW = 1004, /\* 图片质量分数不满足 \*/

NETDEV\_PERSON\_CODE\_PICTURE\_ZOOM\_FAIL = 1005, /\* 图片缩放失败 \*/

NETDEV\_PERSON\_CODE\_INTELLECT\_DISABLE = 1006, /\* 未启用智能 \*/

NETDEV\_PERSON\_CODE\_PICTURE\_TOO\_SMALL = 1007, /\* 导入图片过小 \*/

NETDEV\_PERSON\_CODE\_PICTURE\_TOO\_LARGE = 1008, /\* 导入图片过大 \*/

NETDEV\_PERSON\_CODE\_RESOLUTION\_TOO\_LARGE = 1009, /\* 导入图片分辨率超过1920\*1080 \*/

NETDEV\_PERSON\_CODE\_PICTURE\_NON\_EXISTENT = 1010, /\* 导入图片不存在 \*/

NETDEV\_PERSON\_CODE\_FACE\_ELEMENTS\_LIMIT = 1011, /\* 人脸元素个数已达到上限 \*/

NETDEV\_PERSON\_CODE\_INTELLECT\_MODULE\_MISMATCH = 1012, /\* 智能棒算法模型不匹配 \*/

NETDEV\_PERSON\_CODE\_DOCUMENT\_ID\_INVLID = 1013, /\* 人脸导入库成员证件号非法 \*/

NETDEV\_PERSON\_CODE\_PICTURE\_FORMAT\_ERROR = 1014, /\* 人脸导入库成员图片格式错误 \*/

NETDEV\_PERSON\_CODE\_MONITOR\_DEVICE\_LIMIT = 1015, /\* 通道布控已达设备能力上限 \*/

NETDEV\_PERSON\_CODE\_FACE\_LIBRARY\_LOCKED = 1016, /\* 其它客户端正在进行操作人脸库 \*/

NETDEV\_PERSON\_CODE\_FACE\_LIBRARY\_UPDATING = 1017, /\* 人脸库文件正在更新中 \*/

NETDEV\_PERSON\_CODE\_JSON\_DESERIALIZE\_FAIL = 1018, /\* Json反序列化失败 \*/

NETDEV\_PERSON\_CODE\_BASE64\_DECODE\_FAIL = 1019, /\* Base64解码失败 \*/

NETDEV\_PERSON\_CODE\_PICTURE\_SIZE\_MISMATCH = 1020, /\* 人脸照片，编码后的大小和实际接收到的长度不一致 \*/

NETDEV\_PERSON\_CODE\_DEV\_PROTOCOL\_DIFFER = 1021, /\* 布控任务只能选择相同图片接入协议的设备 \*/

NETDEV\_PERSON\_CODE\_REACH\_LIMIT = 1022, /\* 设备人脸库达到上限 \*/

NETDEV\_PERSON\_CODE\_NO\_SMART = 1023, /\* 无智能棒 \*/

NETDEV\_PERSON\_CODE\_DEV\_BUSY = 1024, /\* 设备正忙 \*/

NETDEV\_PERSON\_CODE\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_PERSON\_RESULT\_CODE\_E;

### 告警联动配置信息枚举

typedef enum tagPersonCompareResult

{

NETDEV\_TYPE\_COMPARE\_SUCCESS = 1, /\* 比对成功 \*/

NETDEV\_TYPE\_COMPARE\_FAILED = 2, /\* 比对失败 \*/

NETDEV\_TYPE\_COMPARE\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_PERSON\_COMPARE\_RESULT\_TYPE\_E;

### 使能联动参数枚举

typedef enum tagNETDEVAlarmActID

{

ALARM\_ACTION\_TYPE\_NVR\_PREVIEW = 0, /\* 联动NVR预览，详见 NETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_BUZZER = 1, /\* 联动蜂鸣器，IPC暂不支持,NVR 详见NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_E\_MAIL = 2, /\* 联动E-Mail，IPC暂不支持，NVR 详见 NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_RECORD = 3, /\* 联动存储，IPC暂不支持，NVR 详见NETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_PRESET = 4, /\* 联动云台预置位，详见NETDEV\_PRESET\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_OUTPUT\_SWITCH = 5, /\* 联动开关量输出，详见 NETDEV\_OUTPUT\_SWITCH\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_SNAP = 6, /\* 联动抓拍，IPC无需填写ActParam ，NVR 详见NETDEV\_CHANNEL\_ACT\_PARAM\_INFO\_S\*/

ALARM\_ACTION\_TYPE\_BOX = 7, /\* 告警弹框，IPC暂不支持，NVR 详见NETDEV\_ENABLED\_ACT\_PARAM\_INFO\_S \*/

ALARM\_ACTION\_TYPE\_CENTER\_RECORD = 8, /\* 联动中心存储，IPC无需填写ActParam，NVR不支持 \*/

ALARM\_ACTION\_TYPE\_START\_LOCAL\_RECORD = 9, /\* 联动启动本地存储，IPC无需填写ActParam，NVR不支持 \*/

ALARM\_ACTION\_TYPE\_STOP\_LOCAL\_RECORD = 10, /\* 联动停止本地存储，IPC无需填写ActParam，NVR不支持 \*/

ALARM\_ACTION\_TYPE\_SNAP\_UP\_FTP = 11, /\* 联动抓拍上传FTP，IPC无需填写ActParam，NVR不支持 \*/

ALARM\_ACTION\_TYPE\_SNAP\_UP\_EMAIL = 12, /\* 联动抓拍上传EMail，IPC无需填写ActParam，NVR不支持 \*/

ALARM\_ACTION\_TYPE\_SNAP\_UP\_FTP\_AND\_EMAIL = 13, /\* 联动抓拍上传FTP和EMail，IPC无需填写ActParam，NVR不支持 \*/

ALARM\_ACTION\_TYPE\_SMART\_SNAP\_UP = 14, /\* 智能联动抓拍上传，IPC无需填写ActParam，NVR不支持 \*/

ALARM\_ACTION\_TYPE\_FACE\_PIC\_SNAP\_UP = 15, /\* 联动人脸小图抓拍上传，IPC无需填写ActParam，NVR不支持 \*/

ALARM\_ACTION\_TYPE\_ALARM\_REPORT = 16, /\* 联动告警上报，IPC无需填写ActParam，NVR不支持 \*/

ALARM\_ACTION\_TYPE\_PTZ\_ZOOM = 17, /\* 联动云台变倍，IPC不支持 \*/

ALARM\_ACTION\_TYPE\_INVALID = 0xff /\* 无效参数 \*/

}NETDEV\_ALARM\_ACT\_ID\_E;

### 开关量状态枚举

typedef enum tagNETDEVRelayOutPutState

{

NETDEV\_BOOLEAN\_STATUS\_ACTIVE = 0, /\* 触发状态 \*/

NETDEV\_BOOLEAN\_STATUS\_INACTIVE = 1 /\* 非触发状态 \*/

}NETDEV\_RELAYOUTPUT\_STATE\_E;

### 布防计划使能枚举

typedef enum tagNETDEVDefencePlanEnable

{

NETDEV\_DEFENCE\_PLAN\_DISABLE = 0, /\* 不使能 \*/

NETDEV\_DEFENCE\_PLAN\_ENABLE = 1 /\* 使能 \*/

}NETDEV\_DEFENCE\_PLAN\_ENABLE\_E;

### 星期枚举

typedef enum tagNETDEVWeekInfo

{

NETDEV\_LAPI\_WEEK\_MONDAY = 1, /\* 星期一 Monday \*/

NETDEV\_LAPI\_WEEK\_TUESDAY = 2, /\* 星期二 Tuesday \*/

NETDEV\_LAPI\_WEEK\_WEDNESDAY = 3, /\* 星期三 Wednesday \*/

NETDEV\_LAPI\_WEEK\_THURSDAY = 4, /\* 星期四 Thursday \*/

NETDEV\_LAPI\_WEEK\_FRIDAY = 5, /\* 星期五 Friday \*/

NETDEV\_LAPI\_WEEK\_SATURDAY = 6, /\* 星期六 Saturday \*/

NETDEV\_LAPI\_WEEK\_SUNDAY = 7, /\* 星期日 Sunday \*/

NETDEV\_LAPI\_WEEK\_HOLIDAY = 8, /\* 假日 Holiday \*/

NETDEV\_LAPI\_WEEK\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_LAPI\_WEEK\_INFO\_E;

### 告警布防类型枚举

typedef enum tagNETDEVArmingType

{

NETDEV\_ARMING\_TYPE\_TIMING = 0, /\* 定时 \*/

NETDEV\_ARMING\_TYPE\_MOTIONDETEC = 1, /\* 动检 \*/

NETDEV\_ARMING\_TYPE\_ALARM = 2, /\* 报警 \*/

NETDEV\_ARMING\_TYPE\_MOTIONDETEC\_AND\_ALARM = 3, /\* 动检和报警 \*/

NETDEV\_ARMING\_TYPE\_MOTIONDETEC\_OR\_ALARM = 4, /\* 动检或报警 \*/

NETDEV\_ARMING\_TYPE\_NO\_PLAN = 5, /\* 无计划 \*/

NETDEV\_ARMING\_TYPE\_EVENT = 10 /\* 事件 \*/

}NETDEV\_ARMING\_TYPE\_E;

**Remarks：**

* 各类告警布防只支持定时类型，录像/抓图计划全支持，IPC只支持定时，不需要带字段，默认是定时；

### 人员库类型枚举

typedef enum tagNETDEVPeopleLibType

{

NETDEV\_PEOPLE\_LIB\_TYPE\_DEFAULT = 0, /\* 默认无效值 \*/

NETDEV\_PEOPLE\_LIB\_TYPE\_BLACKLIST = 1, /\* 黑名单 \*/

NETDEV\_PEOPLE\_LIB\_TYPE\_STRANGER = 2, /\* 灰名单/陌生人 \*/

NETDEV\_PEOPLE\_LIB\_TYPE\_STAFF = 3, /\* 员工 \*/

NETDEV\_PEOPLE\_LIB\_TYPE\_VISITOR = 4, /\* 访客 \*/

NETDEV\_PEOPLE\_LIB\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_PEOPLE\_LIB\_TYPE\_E;

### 人脸布控操作结果错误码枚举

typedef enum tagNETDEVPersonMonitorOptResCode

{

NETDEV\_PERSON\_MONITOR\_CODE\_INIT\_DETECT\_FAIL = 11702, /\* 初始化检测失败 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_FACE\_DETECT\_FAIL = 11703, /\* 人脸检测失败 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_IMAGE\_NOT\_FIND\_FACE = 11704, /\* 图片未检测到人脸 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_JPEG\_PARSE\_FAIL = 11705, /\* jpeg照片解码失败 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_IMAGE\_MASS\_NOT\_ENOUGH = 11706, /\* 人脸图片质量分数不满足 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_IMAGE\_ZOOM\_FAIL = 11707, /\* 图片缩放失败 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_NOT\_START\_SMART = 11708, /\* 未启用智能 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_PICTURE\_TOO\_SMALL = 11709, /\* 导入图片过小 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_CREATE\_FACE\_LIB\_FAIL = 11710, /\* 创建人脸库失败 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_CREATE\_MONITOR\_FAIL = 11711, /\* 创建布控任务失败 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_PICTURE\_TOO\_LARGE = 11714, /\* 导入图片过大 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_RESOLUTION\_TOO\_LARGE = 11715, /\* 导入图片分辨率超过1920\*1080 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_PICTURE\_NON\_EXISTENT = 11716, /\* 导入图片不存在 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_FACE\_ELEMENTS\_LIMIT = 11717, /\* 人脸元素个数已达到上限 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_INTELLECT\_MODULE\_MISMATCH = 11718, /\* 智能棒算法模型不匹配 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_DOCUMENT\_ID\_INVLID = 11719, /\* 人脸导入库成员证件号非法 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_PICTURE\_FORMAT\_ERROR = 11720, /\* 人脸导入库成员图片格式错误 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_MONITOR\_DEVICE\_LIMIT = 11721, /\* 通道布控已达设备能力上限 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_FACE\_LIBRARY\_LOCKED = 11722, /\* 其它客户端正在进行操作人脸库 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_FACE\_LIBRARY\_UPDATING = 11723, /\* 人脸库文件正在更新中 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_JSON\_DESERIALIZE\_FAIL = 11724, /\* Json反序列化失败 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_BASE64\_DECODE\_FAIL = 11725, /\* Base64解码失败 \*/

NETDEV\_PERSON\_MONITOR\_CODE\_PICTURE\_SIZE\_MISMATCH = 11726 /\* 人脸照片，编码后的大小和实际接收到的长度不一致 \*/

}NETDEV\_PERSON\_MONITOR\_OPT\_RES\_CODE\_E;

### 告警抓图类型枚举

typedef enum tagNETDEVAlarmSnapShotType

{

NETDEV\_ALARM\_SNAPSHOT\_MOTION\_DETECT = 0, /\* 移动告警 \*/

NETDEV\_ALARM\_SNAPSHOT\_VIDEO\_LOST = 1, /\* 视频丢失告警 \*/

NETDEV\_ALARM\_SNAPSHOT\_AUDIO\_DETECTION = 2, /\* 声音检测告警 \*/

NETDEV\_ALARM\_SNAPSHOT\_INPUT\_SWITCH = 3, /\* 开关量输入告警 \*/

NETDEV\_ALARM\_SNAPSHOT\_IPC\_OFFLINE = 4, /\* IPC离线 \*/

NETDEV\_ALARM\_SNAPSHOT\_TAMPER\_DETECT = 5, /\* 遮挡检测 \*/

NETDEV\_ALARM\_SNAPSHOT\_CROSS\_LINE = 6, /\* 越界检测，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_INTROSION\_ZONE = 7, /\* 区域入侵，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_FACE\_DETECT = 8, /\* 人脸检测，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_SCENCECHANGE = 9, /\* 场景变更，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_OUT\_FOCUS = 10, /\* 虚焦检测，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_FACE\_MATCH = 11, /\* 人脸识别匹配告警，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_FACE\_NOT\_MATCH = 12, /\* 人脸识别不匹配报警，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_GATHER = 13, /\* 人员聚集，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_REMAIN\_ARTICLE = 14, /\* 物品遗留，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_SMART\_TRACK = 15, /\* 智能跟踪，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_LOITERING\_DETECTOR = 16, /\* 徘徊检测，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_VEHICLE\_MATCH = 17, /\* 车辆识别匹配报警，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_VEHICLE\_NOT\_MATCH = 18, /\* 车辆识别不匹配报警，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_ENTER\_AREA = 19, /\* 进入区域，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_LEAVE\_AREA = 20, /\* 离开区域，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_ARTICLE\_MOVE = 21, /\* 物品搬移，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_DURESS = 22, /\* 挟持报警，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_ANRIDISASSEMBLY = 23, /\* 防拆报警，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_ANRIDISASSEMBLY\_RECOVER = 24, /\* 防拆报警恢复，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_BYPASS\_OPERATION = 25, /\* 防区旁路操作，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_CF\_BJ\_EMBNAM\_ZONE = 26, /\* 安标报警模块.防区报警，仅一体机支持 \*/

NETDEV\_ALARM\_SNAPSHOT\_CONFLAGRATION = 27, /\* 火点告警 \*/

NETDEV\_ALARM\_SNAPSHOT\_HUMAN\_SHAPE\_DETECT = 28, /\* 人形检测告警 \*/

NETDEV\_ALARM\_SNAPSHOT\_INVALID = 0xFFFF /\* 无效值 \*/

}NETDEV\_ALARM\_SNAPSHOT\_TYPE\_E;

### 5ePTZ显示模式枚举

typedef enum tagNETDEVPTZDisplayMode

{

NETDEV\_PTZ\_DISPLAY\_ORIGINAL = 0, /\*原始图像\*/

NETDEV\_PTZ\_DISPLAY\_5PTZ = 1, /\*全景+5ePTZ模式\*/

NETDEV\_PTZ\_DISPLAY\_INVALID = 0xff /\* 无效值\*/

}NETDEV\_PTZ\_DISPLAY\_MODE\_E;

### 鱼眼设备矫正模式枚举

typedef enum tagNetDEVFishEyePtzMode

{

NETDEV\_FISHEYE\_MODE\_ORIGINAL = 0, /\* 原始图像\*/

NETDEV\_FISHEYE\_MODE\_180 = 1, /\* 2\*180度模式\*/

NETDEV\_FISHEYE\_MODE\_360\_1PTZ = 2, /\* 360度+1PTZ模式\*/

NETDEV\_FISHEYE\_MODE\_360\_6PTZ = 3, /\* 360度+6PTZ模式\*/

NETDEV\_FISHEYE\_MODE\_3PTZ = 4, /\* 鱼眼+3PTZ模式\*/

NETDEV\_FISHEYE\_MODE\_MID\_ON\_4PTZ = 5, /\* 鱼眼图像在中间且显示+4PTZ模式\*/

NETDEV\_FISHEYE\_MODE\_MID\_OFF\_4PTZ= 6, /\* 鱼眼图像在中间但不显示+4PTZ模式\*/

NETDEV\_FISHEYE\_MODE\_LEFT\_4PTZ = 7, /\* 鱼眼左边+4PTZ模式\*/

NETDEV\_FISHEYE\_MODE\_8PTZ = 8, /\* 鱼眼+8PTZ模式\*/

NETDEV\_FISHEYE\_MODE\_PANORAMA = 9, /\* 全景模式\*/

NETDEV\_FISHEYE\_MODE\_PR\_3PTZ = 10, /\* 全景+3PTZ模式\*/

NETDEV\_FISHEYE\_MODE\_PR\_4PTZ = 11, /\* 全景+4PTZ模式\*/

NETDEV\_FISHEYE\_MODE\_PR\_8PTZ = 12, /\* 全景+8PTZ模式\*/

NETDEV\_FISHEYE\_MODE\_INVALID = 0xFF /\* 非法值\*/

}NETDEV\_FISHEYE\_PTZ\_MODE\_E;

### 设备安装模式枚举

typedef enum tagNETDEVfInstallMode

{

NETDEV\_INSTALL\_MODE\_TOP = 0, /\* 顶部安装\*/

NETDEV\_INSTALL\_MODE\_BOTTOM = 1, /\* 底部安装\*/

NETDEV\_INSTALL\_MODE\_SIDE = 2, /\* 侧面安装\*/

NETDEV\_INSTALL\_MODE\_INVALID = 0xFF /\* 非法值\*/

}NETDEV\_INSTALL\_MODE\_E;

### 鼠标移动模式枚举

typedef enum tagNETDEVfMouseMoveMode

{

NETDEV\_MOUSE\_MOVE = 0, /\* 鼠标位移\*/

NETDEV\_MOUSE\_LEFT\_BTN\_DOWN = 1, /\* 左键按下\*/

NETDEV\_MOUSE\_LEFT\_BTN\_UP = 2, /\* 左键弹起\*/

NETDEV\_MOUSE\_WHEEL = 3, /\* 滚轮操作\*/

NETDEV\_MOUSE\_INVALID = 0xFF /\* 非法值\*/

}NETDEV\_MOUSE\_MOVE\_MODE\_E;

### 录像类型枚举

typedef enum tagNETDEVRecordType

{

NETDEV\_RECORD\_TYPE\_MANUAL = 0, /\*手动录像\*/

NETDEV\_RECORD\_TYPE\_NORMAL = 1 /\*普通录像\*/

}NETDEV\_RECORD\_TYPE\_E;

### 录像状态枚举

typedef enum tagNETDEVManualRecordStatus

{

NETDEV\_MANUAL\_RECORD\_STATUS\_FREE = 0, /\* 未录像\*/

NETDEV\_MANUAL\_RECORD\_STATUS\_RECORDING = 1, /\* 录像中\*/

NETDEV\_MANUAL\_RECORD\_STATUS\_INVALID = 0xFF /\* 无效命令\*/

}NETDEV\_MANUAL\_RECORD\_STATUS\_E;

### 设备配置命令

typedef enum tagNETDEVCfgCmd

{

NETDEV\_GET\_DEVICECFG = 100, /\* 获取设备信息，

参见#[NETDEV\_DEVICE\_BASICINFO\_S](#_设备基本信息) \*/

NETDEV\_SET\_DEVICECFG = 101, /\* 保留\*/

NETDEV\_GET\_NTPCFG = 110, /\* 获取NTP参数,

参见#[NETDEV\_SYSTEM\_NTP\_INFO\_S](#_NTP参数) \*/

NETDEV\_SET\_NTPCFG = 111, /\* 设置NTP参数,

参见#[NETDEV\_SYSTEM\_NTP\_INFO\_S](#_NTP参数) \*/

NETDEV\_GET\_NTPCFG\_EX = 112, /\* 获取NTP参数(扩展，建议使用),

参见#[NETDEV\_SYSTEM\_NTP\_INFO\_LIST\_S](#_NTP列表) \*/

NETDEV\_SET\_NTPCFG\_EX = 113, /\* 设置NTP参数(扩展，建议使用),

参见#[NETDEV\_SYSTEM\_NTP\_INFO\_LIST\_S](#_NTP列表) \*/

NETDEV\_GET\_STREAMCFG = 120, /\* 获取视频编码参数,

7参见#[NETDEV\_VIDEO\_STREAM\_INFO\_S](#_通道视频流信息) \*/

NETDEV\_SET\_STREAMCFG = 121, /\* 设置视频编码参数,

参见#[NETDEV\_VIDEO\_STREAM\_INFO\_S](#_通道视频流信息) \*/

NETDEV\_GET\_STREAMCFG\_EX = 122, /\* 获取视频编码参数(扩展，建议使用),

参见#[NETDEV\_VIDEO\_STREAM\_INFO\_LIST\_S](#_视频流信息列表) \*/

NETDEV\_SET\_STREAMCFG\_EX = 123, /\* 设置视频编码参数(扩展，建议使用),

参见#[NETDEV\_VIDEO\_STREAM\_INFO\_LIST\_S](#_视频流信息列表) \*/

NETDEV\_GET\_VIDEOMODECFG = 124, /\* 获取图像采集制式，

参见#[NETDEV\_VIDEO\_MODE\_INFO\_S](#_视频制式能力) \*/

NETDEV\_SET\_VIDEOMODECFG = 125, /\* 设置图像采集制式，

参见#[NETDEV\_VIDEO\_MODE\_INFO\_S](#_视频制式能力) \*/

NETDEV\_GET\_OSDCFG = 140, /\* 获取OSD配置信息,

参见#[NETDEV\_VIDEO\_OSD\_CFG\_S](#_通道OSD的基本属性信息) \*/

NETDEV\_SET\_OSDCFG = 141, /\* 设置OSD配置信息,

参见#[NETDEV\_VIDEO\_OSD\_CFG\_S](#_通道OSD的基本属性信息) \*/

NETDEV\_GET\_OSDCFG\_EX = 142, /\* 获取OSD配置列表,

参见#[NETDEV\_CHL\_OSD\_CFG\_LIST\_S](#_通道OSD配置列表) \*/

NETDEV\_SET\_OSDCFG\_EX = 143, /\* 设置OSD配置列表,

参见#[NETDEV\_CHL\_OSD\_CFG\_LIST\_S](#_通道OSD配置列表) \*/

NETDEV\_GET\_OSD\_CONTENT\_CFG = 144, /\* 获取OSD配置信息(扩展，建议使用),

参见#[NETDEV\_OSD\_CONTENT\_S](#_通道OSD所有内容) \*/

NETDEV\_SET\_OSD\_CONTENT\_CFG = 145, /\* 设置OSD配置信息(扩展，建议使用),

参见#[NETDEV\_OSD\_CONTENT\_S](#_通道OSD所有内容) \*/

NETDEV\_GET\_OSD\_CONTENT\_STYLE\_CFG = 146, /\* 获取OSD内容样式,

参见#[NETDEV\_OSD\_CONTENT\_STYLE\_S](#_通道OSD内容样式) \*/

NETDEV\_SET\_OSD\_CONTENT\_STYLE\_CFG = 147, /\* 设置OSD内容样式,

参见#[NETDEV\_OSD\_CONTENT\_STYLE\_S](#_通道OSD内容样式) \*/

NETDEV\_GET\_ALARM\_OUTPUTCFG = 150, /\* 获取开关量输出配置信息,

参见#[NETDEV\_ALARM\_OUTPUT\_LIST\_S](#_所有告警开关量输入信息) \*/

NETDEV\_SET\_ALARM\_OUTPUTCFG = 151, /\* 设置开关量输出配置信息,

参见#[NETDEV\_ALARM\_OUTPUT\_LIST\_S](#_所有告警开关量输入信息) \*/

NETDEV\_TRIGGER\_ALARM\_OUTPUT = 152, /\* 触发开关量,

参见#[NETDEV\_TRIGGER\_ALARM\_OUTPUT\_S](#_触发开关量输入告警) \*/

NETDEV\_GET\_ALARM\_INPUTCFG = 153, /\* 获取开关量输入数量,

参见#[NETDEV\_ALARM\_INPUT\_LIST\_S](#_所有告警开关量输入信息_1) \*/

NETDEV\_GET\_MANUAL\_ALARM\_CFG = 154, /\* 获取手动告警开关量配置信息，

参见#[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, /\* 设置手动告警开关量配置信息，

参见#[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, /\* 获取所有输入开关量配置，

参见#[NETDEV\_INPUT\_SWITCH\_INFO\_LIST\_S](#_输入开关量告警配置信息列表) \*/

NETDEV\_GET\_IMAGECFG = 160, /\* 获取图像配置信息,

参见#[NETDEV\_IMAGE\_SETTING\_S](#_设备图像设置) \*/

NETDEV\_SET\_IMAGECFG = 161, /\* 设置图像配置信息,

参见#[NETDEV\_IMAGE\_SETTING\_S](#_设备图像设置) \*/

NETDEV\_GET\_IMAGE\_EXPOSURE = 162, /\* 获取通道的图像曝光参数,

参见#[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数)\*/

NETDEV\_SET\_IMAGE\_EXPOSURE = 163, /\* 设置通道的图像曝光参数,

参见#[NETDEV\_IMAGE\_EXPOSURE\_S](#_图像曝光参数) \*/

NETDEV\_GET\_IMAGE\_LAMP\_CTRLCFG = 164, /\* 获取补光灯参数,

参见#[NETDEV\_LAMP\_CTRL\_INFO\_S](#_补光灯信息) \*/

NETDEV\_SET\_IMAGE\_LAMP\_CTRLCFG = 165, /\* 设置补光灯参数,

参见#[NETDEV\_LAMP\_CTRL\_INFO\_S](#_补光灯信息) \*/

NETDEV\_GET\_IMAGE\_WHITE\_BALANCE = 166, /\* 获取图像白平衡参数,

参见# [NETDEV\_WHITE\_BALANCE\_INFO\_S](#_白平衡信息) \*/

NETDEV\_SET\_IMAGE\_WHITE\_BALANCE = 167, /\* 设置图像白平衡参数,

参见#  [NETDEV\_WHITE\_BALANCE\_INFO\_S](#_白平衡信息) \*/

NETDEV\_SET\_IMAGE\_PARAM\_RESET = 168, /\* 设置图像参数默认值\*/

NETDEV\_GET\_NETWORKCFG = 170, /\* 获取网络配置信息,

参见#[NETDEV\_NETWORKCFG\_S](#_网络配置信息) \*/

NETDEV\_SET\_NETWORKCFG = 171, /\* 设置网络配置信息,

参见#[NETDEV\_NETWORKCFG\_S](#_网络配置信息) \*/

NETDEV\_GET\_VPN\_CFG = 172, /\* 获取VPN客户端指定模式信息

参见#[NETDEV\_VPN\_CLIENT\_INFO\_S](#_VPN客户端指定模式信息) \*/

NETDEV\_SET\_VPN\_CFG = 173, /\* 设置VPN客户端指定模式信息

参见#[NETDEV\_VPN\_CLIENT\_INFO\_S](#_VPN客户端指定模式信息) \*/

NETDEV\_SET\_DNS\_CFG = 174, /\* 设置DNS地址信息

参见#[NETDEV\_DNS\_INFO\_S](#_设备DNS信息) \*/

NETDEV\_GET\_DNS\_CFG = 175, /\* 获取DNS地址信息

参见#[NETDEV\_DNS\_INFO\_S](#_设备DNS信息) \*/

NETDEV\_SET\_NETWORK\_CARDS = 176, /\* 设置设备所有网卡的配置信息

参见#[NETDEV\_NETWORK\_CARD\_INFO\_S](#_网卡信息) \*/

NETDEV\_GET\_NETWORK\_CARDS = 177, /\* 获取设备所有网卡的配置信息

参见#[NETDEV\_NETWORK\_CARD\_INFO\_S](#_网卡信息) \*/

NETDEV\_SET\_NETWORK\_PORTS = 178, /\* 设置设备网络端口信息

参见#[NETDEV\_NETWORK\_PORTS\_INFO\_S](#_端口信息) \*/

NETDEV\_GET\_NETWORK\_PORTS = 179, /\* 获取设备网络端口信息

参见#[NETDEV\_NETWORK\_PORTS\_INFO\_S](#_端口信息) \*/

NETDEV\_GET\_PRIVACYMASKCFG = 180, /\* 获取隐私遮盖配置信息,

参见#[NETDEV\_PRIVACY\_MASK\_CFG\_S](#_隐私遮盖配置信息) \*/

NETDEV\_SET\_PRIVACYMASKCFG = 181, /\* 设置隐私遮盖配置信息,

参见#[NETDEV\_PRIVACY\_MASK\_CFG\_S](#_隐私遮盖配置信息) \*/

NETDEV\_DELETE\_PRIVACYMASKCFG = 182, /\* 删除隐私遮盖配置信息 \*/

NETDEV\_GET\_TAMPERALARM = 190, /\* 获取遮挡检测告警信息

参见#[NETDEV\_TAMPER\_ALARM\_INFO\_S](#_遮挡检测分析信息) \*/

NETDEV\_SET\_TAMPERALARM = 191, /\* 设置遮挡检测告警信息

参见#[NETDEV\_TAMPER\_ALARM\_INFO\_S](#_遮挡检测分析信息) \*/

NETDEV\_GET\_MOTIONALARM = 200, /\* 获取运动检测告警信息

参见#[NETDEV\_MOTION\_ALARM\_INFO\_S](#_运动检测分析信息) \*/

NETDEV\_SET\_MOTIONALARM = 201, /\* 设置运动检测告警信息

参见#[NETDEV\_MOTION\_ALARM\_INFO\_S](#_运动检测分析信息) \*/

NETDEV\_GET\_CROSSLINEALARM = 202, /\* 获取越界检测告警信息

参见[NETDEV\_CROSS\_LINE\_ALARM\_INFO\_S](#_越界检测分析信息) \*/

NETDEV\_SET\_CROSSLINEALARM = 203, /\* 设置越界检测告警信息

参见[NETDEV\_CROSS\_LINE\_ALARM\_INFO\_S](#_越界检测分析信息) \*/

NETDEV\_GET\_INTRUSIONALARM = 204, /\* 获取入侵检测告警信息

参见[NETDEV\_INTRUSION\_ALARM\_INFO\_S](#_入侵检测分析信息) \*/

NETDEV\_SET\_INTRUSIONALARM = 205, /\* 设置入侵检测告警信息

参见[NETDEV\_INTRUSION\_ALARM\_INFO\_S](#_入侵检测分析信息) \*/

NETDEV\_GET\_DISKSINFO = 210, /\* 获取硬盘信息

参见# [NETDEV\_DISK\_INFO\_LIST\_S](#_硬盘信息列表) \*/

NETDEV\_SET\_DISKSINFO = 211, /\* 设置硬盘信息，保留\*/

NETDEV\_GET\_FOCUSINFO = 230, /\* 获取聚焦信息

参见#[NETDEV\_FOCUS\_INFO\_S](#_聚焦信息) \*/

NETDEV\_SET\_FOCUSINFO = 231, /\* 设置聚焦信息

参见#[NETDEV\_FOCUS\_INFO\_S](#_聚焦信息) \*/

NETDEV\_GET\_IRCUTFILTERINFO = 232, /\* 获取昼夜模式信息

参见#[NETDEV\_IRCUT\_FILTER\_INFO\_S](#_昼夜模式信息_1) \*/

NETDEV\_SET\_IRCUTFILTERINFO = 233, /\* 设置昼夜模式信息

参见#[NETDEV\_IRCUT\_FILTER\_INFO\_S](#_昼夜模式信息_1) \*/

NETDEV\_GET\_DEFOGGINGINFO = 234, /\* 获取透雾模式信息

参见#[NETDEV\_DEFOGGING\_INFO\_S](#_透雾信息) \*/

NETDEV\_SET\_DEFOGGINGINFO = 235, /\* 设置透雾模式信息

参见#[NETDEV\_DEFOGGING\_INFO\_S](#_透雾信息) \*/

NETDEV\_SET\_CHL\_IMAGE\_ENHANCE = 236, /\* 设置通道的图像增强

参数 参见[NETDEV\_IMAGE\_ENHANCE\_S](#_指定通道的图像增强参数) \*/

NETDEV\_GET\_CHL\_IMAGE\_ENHANCE = 237, /\* 获取通道的图像增强

参数参见[NETDEV\_IMAGE\_ENHANCE\_S](#_指定通道的图像增强参数) \*/

NETDEV\_GET\_AUDIOIN\_CFG = 240, /\* 获取音频输入参数配置信息

参见#[NETDEV\_AUDIO\_INPUT\_CFG\_INFO\_S](#_音频输入参数配置信息) \*/

NETDEV\_SET\_AUDIOIN\_CFG = 241, /\* 设置音频输入参数配置信息

参见#[NETDEV\_AUDIO\_INPUT\_CFG\_INFO\_S](#_音频输入参数配置信息) \*/

NETDEV\_GET\_DST\_CFG = 260, /\* 获取夏令时配置信息

参见#[NETDEV\_DST\_CFG\_S](#_夏令时配置信息) \*/

NETDEV\_SET\_DST\_CFG = 261, /\* 设置夏令时配置信息

参见#[NETDEV\_DST\_CFG\_S](#_夏令时配置信息) \*/

NETDEV\_GET\_RECORDPLANINFO = 270, /\* 获取录像计划配置信息支持NVR VMS 参见[NETDEV\_RECORD\_PLAN\_CFG\_INFO\_S](#_录像计划配置信息)\*/

NETDEV\_SET\_RECORDPLANINFO = 271, /\* 设置录像计划配置信息支持NVR VMS 参见#[NETDEV\_RECORD\_PLAN\_CFG\_INFO\_S](#_录像计划配置信息) \*/

NETDEV\_SET\_SNAPSHOT\_CFG = 272, /\* 设置指定通道的抓图

参数参见#[NETDEV\_VIDEO\_SNAPSHOT\_S](#_视频输入通道的抓图信息) \*/

NETDEV\_GET\_SNAPSHOT\_CFG = 273, /\* 获取指定通道的抓图

参数参见#[NETDEV\_VIDEO\_SNAPSHOT\_S](#_视频输入通道的抓图信息)\*/

NETDEV\_GET\_MOTIONDETECTION\_WEEKPLAN = 280, /\* 获取移动侦测布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_MOTIONDETECTION\_WEEKPLAN = 281, /\* 设置移动侦测布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_CROSSLINEDETECTION\_WEEKPLAN = 282, /\* 获取越界检测布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_CROSSLINEDETECTION\_WEEKPLAN = 283, /\* 设置越界检测布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_INTRUSIONDETECTION\_WEEKPLAN = 284, /\* 获取区域入侵布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_INTRUSIONDETECTION\_WEEKPLAN = 285, /\* 设置区域入侵布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_AUDIODETECTION\_WEEKPLAN = 286, /\* 获取音频检测布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_AUDIODETECTION\_WEEKPLAN = 287, /\* 设置音频检测布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_OBJTRACK\_WEEKPLAN = 288, /\* 获取目标跟踪布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_OBJTRACK\_WEEKPLAN = 289, /\* 设置目标跟踪布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_VIDEOLOSS\_WEEKPLAN = 290, /\* 获取视频丢失的布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_VIDEOLOSS\_WEEKPLAN = 291, /\* 设置视频丢失的布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_TAMPERDETECTION\_WEEKPLAN = 292, /\* 获取遮挡检测布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_TAMPERDETECTION\_WEEKPLAN = 293, /\* 设置遮挡检测布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_INPUTSWITCHES\_WEEKPLAN = 294, /\* 获取输入开关量布防计划,

参见#[NETDEV\_SWITCH\_WEEK\_PLAN\_INFO\_S](#_开关量计划（周）配置) \*/

NETDEV\_SET\_INPUTSWITCHES\_WEEKPLAN = 295, /\* 设置输入开关量布防计划,

参见#[NETDEV\_SWITCH\_WEEK\_PLAN\_INFO\_S](#_开关量计划（周）配置) \*/

NETDEV\_GET\_OUTPUTSWITCHES\_WEEKPLAN = 296, /\* 获取输出开关量布防计划,

参见#[NETDEV\_SWITCH\_WEEK\_PLAN\_INFO\_S](#_开关量计划（周）配置) \*/

NETDEV\_SET\_OUTPUTSWITCHES\_WEEKPLAN = 297, /\* 设置输出开关量布防计划,

参见# [NETDEV\_SWITCH\_WEEK\_PLAN\_INFO\_S](#_开关量计划（周）配置) \*/

NETDEV\_GET\_DEFOCUSDETECTION\_WEEKPLAN = 298, /\* 获取虚焦检测布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_DEFOCUSDETECTION\_WEEKPLAN = 299, /\* 设置虚焦检测布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_SCENECHANGE\_WEEKPLAN = 300, /\* 获取场景变更布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_SCENECHANGE\_WEEKPLAN = 301, /\* 设置场景变更布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_FACEDETECTION\_WEEKPLAN = 302, /\* 获取人脸检测布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_SET\_FACEDETECTION\_WEEKPLAN = 303, /\* 设置人脸检测布防计划,

参见#[NETDEV\_WEEK\_PLAN\_INFO\_S](#_计划（周）配置) \*/

NETDEV\_GET\_RECORD\_STATUS = 320, /\* 获取录像状态信息

参见#[NETDEV\_RECORD\_STATUS\_LIST\_S](#_录像状态信息列表) \*/

NETDEV\_GET\_SERIAL\_CFG = 330, /\* 获取串口配置

参见# [NETDEV\_SERIAL\_PORTS\_S](#_串口信息) \*/

NETDEV\_SET\_SERIAL\_CFG = 331, /\* 设置串口配置

参见# [NETDEV\_SERIAL\_PORTS\_S](#_串口信息) \*/

NETDEV\_GET\_AUDIO\_DECODE\_STATUS = 340, /\* 获取通道所有随路音频解码状态

参见#[NETDEV\_AUDIO\_DECODE\_STATUS\_LIST\_S](#_随路音频状态信息列表) \*/

NETDEV\_SET\_AUDIO\_DECODE\_STATUS = 341, /\* 设置通道所有随路音频解码状态

参见#[NETDEV\_AUDIO\_DECODE\_STATUS\_LIST\_S](#_随路音频状态信息列表) \*/

NETDEV\_GET\_VIDEO\_LOSS\_RULE\_INFO = 400, /\* 获取视频丢失告警的配置信息

参见#[NETDEV\_VIDEO\_LOSS\_RULE\_INFO\_S](#_视频丢失告警的配置信息) \*/

NETDEV\_SET\_VIDEO\_LOSS\_RULE\_INFO = 401, /\* 设置视频丢失告警的配置信息

参见#[NETDEV\_VIDEO\_LOSS\_RULE\_INFO\_S](#_视频丢失告警的配置信息) \*/

NETDEV\_GET\_VIDEO\_LOSS\_LINKAGE\_ACTIONS = 410, /\* 获取视频丢失告警的联动动作

参见#[NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表) \*/

NETDEV\_SET\_VIDEO\_LOSS\_LINKAGE\_ACTIONS = 411, /\* 设置视频丢失告警的联动动作

参见#[NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表) \*/

NETDEV\_GET\_TAMPER\_DETECTION\_RULE\_INFO = 420, /\* 获取视频输入通道的遮挡检测告警的配置参见#[NETDEV\_TAMPER\_DETECTION\_RULE\_INFO\_S](#_遮挡检测告警的配置信息) \*/

NETDEV\_SET\_TAMPER\_DETECTION\_RULE\_INFO = 421, /\* 设置视频输入通道的遮挡检测告警的配置参见#[NETDEV\_TAMPER\_DETECTION\_RULE\_INFO\_S](#_遮挡检测告警的配置信息) \*/

NETDEV\_GET\_TAMPER\_DETECTION\_LINKAGE\_ACTIONS = 430, /\* 获取遮挡检测告警的联动动作

参见# [NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表) \*/

NETDEV\_SET\_TAMPER\_DETECTION\_LINKAGE\_ACTIONS = 431, /\* 设置遮挡检测告警的联动动作参见# [NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表) \*/

NETDEV\_GET\_TEMP\_DETECTION\_RULE\_INFO = 432, /\* 获取视频输入通道的温度检测规则配置参见#[NETDEV\_TEMP\_DETECTION\_RULE\_INFO\_S](#_温度检测告警的配置信息) \*/

NETDEV\_SET\_TEMP\_DETECTION\_RULE\_INFO = 433, /\* 设置视频输入通道的温度检测规则配置参见# [NETDEV\_TEMP\_DETECTION\_RULE\_INFO\_S](#_温度检测告警的配置信息) \*/

NETDEV\_GET\_TEMP\_DETECTION\_BLACKBODY\_INFO = 434, /\* 获取视频输入通道的温度检测黑体参数参见# [NETDEV\_TEMP\_DETECTION\_BLACKBODY\_INFO\_S](#_温度检测黑体信息结构体) \*/

NETDEV\_SET\_TEMP\_DETECTION\_BLACKBODY\_INFO = 435, /\* 设置视频输入通道的温度检测黑体参数参见# [NETDEV\_TEMP\_DETECTION\_BLACKBODY\_INFO\_S](#_温度检测黑体信息结构体) \*/

NETDEV\_GET\_TEMP\_DETECTION\_CORRECT\_INFO = 436, /\* 获取视频输入通道的温度检测修正参数参见# [NETDEV\_TEMP\_DETECTION\_CORRECT\_INFO\_S](#_温度检测矫正信息结构体) \*/

NETDEV\_SET\_TEMP\_DETECTION\_CORRECT\_INFO = 437, /\* 设置视频输入通道的温度检测修正参数参见# [NETDEV\_TEMP\_DETECTION\_CORRECT\_INFO\_S](#_温度检测矫正信息结构体) \*/

NETDEV\_GET\_TEMP\_DETECTION\_UNIT\_INFO = 438, /\* 获取视频输入通道的温度检测温度单位参见# [NETDEV\_TEMP\_DETECTION\_UNIT\_INFO\_S](#_温度检测单位信息结构体) \*/

NETDEV\_SET\_TEMP\_DETECTION\_UNIT\_INFO = 439, /\* 设置视频输入通道的温度检测温度单位参见# [NETDEV\_TEMP\_DETECTION\_UNIT\_INFO\_S](#_温度检测单位信息结构体) \*/

NETDEV\_GET\_MOTION\_DETECTION\_AREA\_TYPE = 440, /\* 获取视频输入通道运动检测的区域类型参见#[NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_S](#_运动检测区域类型信息) \*/

NETDEV\_SET\_MOTION\_DETECTION\_AREA\_TYPE = 441, /\* 设置视频输入通道运动检测的区域类型参见#[NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_S](#_运动检测区域类型信息) \*/

NETDEV\_GET\_MOTION\_DETECTION\_AREA\_GRID\_INFO = 442, /\* 获取视频输入通道运动检测的所有宏块区域信息参见#[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, /\* 设置视频输入通道运动检测的所有宏块区域信息

参见#[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, /\* 获取视频输入通道运动检测中的所有矩形区域信息

参见#[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, /\* 设置视频输入通道运动检测中的所有矩形区域信息

参见#[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, /\* 获取运动检测告警的联动动作参见#[NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表)\*/

NETDEV\_SET\_MOTION\_DETECTION\_LINKAGE\_ACTIONS = 451, /\* 设置运动检测告警的联动动作参见#[NETDEV\_LINKAGE\_ACTION\_LIST\_S](#_布控任务联动动作列表) \*/

NETDEV\_GET\_INPUT\_SWITCHES\_LINKAGE\_ACTIONS = 460, /\* 获取输入开关量告警的联动动作参见#[NETDEV\_SWITCH\_LINKAGE\_ACTION\_S](#_开关量布控任务联动动作) \*/

NETDEV\_SET\_INPUT\_SWITCHES\_LINKAGE\_ACTIONS = 461, /\* 设置输入开关量告警的联动动作参见#[NETDEV\_SWITCH\_LINKAGE\_ACTION\_S](#_开关量布控任务联动动作) \*/

NETDEV\_GET\_RAID\_STATUS = 470, /\* 获取阵列状态参见#[NETDEV\_RAID\_STATUS\_S](#_阵列状态信息) \*/

NETDEV\_GET\_RAID\_STORAGE\_CONTAINER\_INFO\_LIST = 471, /\* 先使用

NETDEV\_GET\_RAID\_STATUS命令获取阵列状态，阵列状态使能时，获取存储容器信息列表

参见#[NETDEV\_HDD\_INFO\_LIST\_S](#_磁盘信息列表) \*/

NETDEV\_GET\_STORAGE\_CONTAINER\_INFO\_LIST = 472, /\* 先使用

NETDEV\_GET\_RAID\_STATUS命令获取阵列状态，阵列状态不使能时，获取存储容器信息列表

参见#[NETDEV\_STORAGE\_CONTAINER\_INFO\_LIST\_S](#_存储容器信息列表) \*/

NETDEV\_GET\_HDD\_SMART\_INFO = 473, /\* 获取指定硬盘的Smart检测信息参见#[NETDEV\_HDD\_SMART\_INFO\_S](#_硬盘Smart信息) \*/

NETDEV\_SET\_HDD\_SMART\_INFO = 474, /\* 设置指定硬盘的Smart检测信息参见#[NETDEV\_HDD\_SMART\_CHECK\_INFO\_S](#_硬盘Smart检测信息) \*/

NETDEV\_GET\_RAID\_STATUS\_V30 = 475, /\* 获取阵列状态支持VMS/NVR 参见#[NETDEV\_RAID\_STATUS\_S](#_阵列状态信息) \*/

NETDEV\_SET\_RAID\_STATUS\_V30 = 476, /\* 设置阵列状态支持VMS/NVR 参见#[NETDEV\_RAID\_STATUS\_S](#_阵列状态信息) \*/

NETDEV\_GET\_CLOUD\_CONVENIENT\_ACCS\_STATUS = 480, /\* 获取

Ezcloud云端下设备便捷添加模式使能状态使能状态0：关闭1：开启\*/

NETDEV\_SET\_CLOUD\_CONVENIENT\_ACCS\_STATUS = 481, /\* 设置

Ezcloud云端下设备便捷添加模式使能状态使能状态0：关闭1：开启\*/

NETDEV\_GET\_PHOTO\_SERVER\_INFO = 482, /\* 获取照片接收服务器配置信息参见# [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, /\* 设置照片接收服务器配置参见# [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, /\* 获取一体

机增强配置信息,参见# [NETDEV\_ENHANCED\_CONFIG\_INFO\_S](#_功能增强配置信息) \*/

NETDEV\_SET\_ENHANCED\_CONFIG = 485, /\* 设置一体机增强配置信息,参见# [NETDEV\_ENHANCED\_CONFIG\_INFO\_S](#_功能增强配置信息) \*/

NETDEV\_GET\_INTELLIGENT\_SERVER\_INFO = 486, /\* 获取智能服务器配置参见# [NETDEV\_DATA\_SERVER\_LIST](#_智能服务器配置列表) \*/

NETDEV\_SET\_INTELLIGENT\_SERVER\_INFO = 487, /\* 设置智能服务器配置参见# [NETDEV\_DATA\_SERVER\_LIST](#_智能服务器配置列表) \*/

NETDEV\_GET\_MOTION\_INTERVAL\_INFO = 490, /\* 查询指定视频输入通道运动检测告警的报警间隔参数参见[NETDEV\_MOTION\_INTERVAL\_INFO\_S](#_运动检测告警参数) \*/

NETDEV\_SET\_MOTION\_INTERVAL\_INFO = 491, /\* 设置指定视频输入通道运动检测告警的报警间隔参数参见[NETDEV\_MOTION\_INTERVAL\_INFO\_S](#_运动检测告警参数) \*/

NETDEV\_GET\_MANAGER\_SERVER\_INFO = 492, /\* 获取管理服务器配置参见# [NETDEV\_MANAGER\_SERVER\_LIST\_S](#_管理服务器配置列表) \*/

NETDEV\_SET\_MANAGER\_SERVER\_INFO = 493, /\* 设置管理服务器配置参见# [NETDEV\_MANAGER\_SERVER\_LIST\_S](#_管理服务器配置列表) \*/

NETDEV\_GET\_VISIBLE\_RANGE\_INFO = 494, /\* 获取可视域参数信息参见# [NETDEV\_VISIBLE\_RANGE\_INFO\_S](#_可视域参数信息) \*/

NETDEV\_SET\_VISIBLE\_RANGE\_INFO = 495, /\* 设置可视域参数信息参见# [NETDEV\_VISIBLE\_RANGE\_INFO\_S](#_可视域参数信息) \*/

NETDEV\_GET\_RESET\_PWD\_VERTIFY\_INFO = 496, /\* 获取找回用户密码的验证信息参见# [NETDEV\_RESET\_PWD\_VERTIFY\_INFO\_S](#_验证信息) \*/

NETDEV\_SET\_RESET\_PWD\_VERTIFY\_INFO = 497, /\* 设置找回用户密码的验证信息参见# [NETDEV\_RESET\_PWD\_VERTIFY\_INFO\_S](#_验证信息)\*/

NETDEV\_GET\_CHL\_STORAGE\_QUOTA\_INFO = 498, /\* 获取通道的具体配额信息参见#[NETDEV\_STORAGE\_QUOTA\_INFO\_S](#_存储配额信息) \*/

NETDEV\_SET\_CHL\_STORAGE\_QUOTA\_INFO = 499, /\* 设置通道的具体配额信息参见#[NETDEV\_STORAGE\_QUOTA\_INFO\_S](#_存储配额信息)  \*/

NETDEV\_GET\_CUSTOM\_PROTOCOL\_INFO = 501, /\* 获取自定义协议配置信息参见#[NETDEV\_CUSTON\_PROTOCOL\_INFO\_S](#_自定义协议信息) \*/

NETDEV\_SET\_CUSTOM\_PROTOCOL\_INFO = 502, /\* 设置自定义协议配置信息参见# [NETDEV\_CUSTON\_PROTOCOL\_INFO\_S](#_自定义协议信息) \*/

NETDEV\_GET\_CUSTOM\_PROTOCOL\_INFOLIST = 503, /\* 获取自定义协议配置信息列表参见#[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, /\* 获取通道存储策略信息参见#[NETDEV\_STORAGE\_STRATEGY\_S](#_存储策略信息) \*/

NETDEV\_SET\_CHL\_STORAGE\_STRATEGY = 505, /\* 设置通道存储策略信息参见# [NETDEV\_STORAGE\_STRATEGY\_S](#_存储策略信息) \*/

NETDEV\_SET\_BACKFOCUS = 506, /\* 调节后焦仅IPC支持参见# [NETDEV\_BACKFOCUS\_INFO\_S](#_后焦调节参数信息) \*/

NETDEV\_GET\_SMART\_ATTR\_MONITOR = 508, /\* 获取智能属性配置，参见# [NETDEV\_SMART\_ATTR\_CFG\_S](#_智能属性配置) \*/

NETDEV\_SET\_SMART\_ATTR\_MONITOR = 509, /\* 设置智能属性配置，参见# [NETDEV\_SMART\_ATTR\_CFG\_S](#_智能属性配置) \*/

NETDEV\_GET\_IMAGE\_CORRECT\_PARAM = 510, /\* 获取图片矫正参数参见# [NETDEV\_IMAGE\_CORRECT\_PARAM\_S](#_图片矫正参数) \*/

NETDEV\_GET\_DEVICE\_TIME\_SYNCMODE = 512, /\* 查询设备时间同步方式，参见[NETDEV\_TIME\_SYNCMODE\_S](#_设备时间同步方式) \*/

NETDEV\_SET\_DEVICE\_TIME\_SYNCMODE = 513, /\* 设置设备时间同步方式，参见[NETDEV\_TIME\_SYNCMODE\_S](#_设备时间同步方式) \*/

NETDEV\_GET\_IMAGE\_SCENE\_AUTO\_SWITCH\_INFO = 514, /\* 查询指定视频输入通道场景自动切换开关信息参见[NETDEV\_SCENE\_AUTO\_SWITCH\_INFO\_S](#_场景自动切换开关信息) \*/

NETDEV\_SET\_IMAGE\_SCENE\_AUTO\_SWITCH\_INFO = 515, /\* 设置指定视频输入通道场景自动切换开关信息参见[NETDEV\_SCENE\_AUTO\_SWITCH\_INFO\_S](#_场景自动切换开关信息) \*/

NETDEV\_GET\_IMAGE\_CURRENT\_SCENE\_TEMPLATE\_INFO = 516, /\* 查询指定视频输入通道当前使用的场景模板参见[NETDEV\_SCENE\_TEMPLATE\_INFO\_S](#_场景模板信息) \*/

NETDEV\_SET\_IMAGE\_CURRENT\_SCENE\_TEMPLATE\_INFO = 517, /\* 设置指定视频输入通道当前使用的场景模板参见[NETDEV\_SCENE\_TEMPLATE\_INFO\_S](#_场景模板信息) \*/

NETDEV\_GET\_IMAGE\_DEFAULT\_SCENE\_TEMPLATE\_INFO = 518, /\* 查询指定视频输入通道的默认场景模板参见[NETDEV\_SCENE\_TEMPLATE\_INFO\_S](#_场景模板信息) \*/

NETDEV\_SET\_IMAGE\_DEFAULT\_SCENE\_TEMPLATE\_INFO = 519, /\* 设置指定视频输入通道的默认场景模板参见[NETDEV\_SCENE\_TEMPLATE\_INFO\_S](#_场景模板信息) \*/

NETDEV\_GET\_IMAGE\_ALL\_SCENE\_INFO = 520, /\* 查询指定视频输入通道的所有场景的信息参见[NETDEV\_SCENE\_INFO\_LIST\_S](#_所有场景信息) \*/

NETDEV\_SET\_IMAGE\_ALL\_SCENE\_INFO = 521, /\* 设置指定视频输入通道的所有场景的信息参见[NETDEV\_SCENE\_INFO\_LIST\_S](#_所有场景信息) \*/

NETDEV\_GET\_IMAGE\_SPECIFIC\_SCENE\_INFO = 522, /\* 查询指定视频输入通道的某个指定场景模板的场景信息参见[NETDEV\_SCENE\_INFO\_S](#_场景信息) \*/

NETDEV\_SET\_IMAGE\_SPECIFIC\_SCENE\_INFO = 523, /\* 设置指定视频输入通道的某个指定场景模板的场景信息参见[NETDEV\_SCENE\_INFO\_S](#_场景信息) \*/

NETDEV\_GET\_ATTR\_COLLECT\_INFO = 524, /\* 获取属性采集信息参见#[NETDEV\_ATTR\_COLLECT\_INFO\_S](#_属性采集信息) \*/

NETDEV\_SET\_ATTR\_COLLECT\_INFO = 525, /\* 设置属性采集信息参见# [NETDEV\_ATTR\_COLLECT\_INFO\_S](#_属性采集信息) \*/

NETDEV\_GET\_SYS\_TIMECFG = 526, /\* 获取设备系统时间配置参数详见[NETDEV\_TIME\_CFG\_V30\_S](#_系统时间配置) 当前只显控支持URL不需要通道号时，通道号需下发xFF \*/

NETDEV\_SET\_SYS\_TIMECFG = 527, /\* 设置设备系统时间配置参数详见[NETDEV\_TIME\_CFG\_V30\_S](#_系统时间配置) 当前只显控支持URL不需要通道号时，通道号需下发NETDEV\_INVALID\_CHANNEL\_ID \*/

NETDEV\_GET\_NTPCFG\_V30 = 528, /\* 获取NTP服务器全部信息参数详见[NETDEV\_SYSTEM\_NTP\_INFO\_LIST\_S](#_NTP列表) 当前只显控支持URL不需要通道号时，通道号需下发NETDEV\_INVALID\_CHANNEL\_ID \*/

NETDEV\_SET\_NTPCFG\_V30 = 529, /\* 设置NTP服务器全部信息参数详见[NETDEV\_SYSTEM\_NTP\_INFO\_LIST\_S](#_NTP列表) 当前只显控支持URL不需要通道号时，通道号需下发NETDEV\_INVALID\_CHANNEL\_ID \*/

NETDEV\_GET\_DEVICE\_TIME\_SYNCMODE\_EX = 530, /\* 查询设备时间同步方式 参数详见[NETDEV\_TIME\_SYNCMODE\_S](#_设备时间同步方式) 当前只显控支持URL不需要通道号时，通道号需下发NETDEV\_INVALID\_CHANNEL\_ID \*/

NETDEV\_SET\_DEVICE\_TIME\_SYNCMODE\_EX = 531, /\* 设置设备时间同步方式 参数详见[NETDEV\_TIME\_SYNCMODE\_S](#_设备时间同步方式) 当前只显控支持URL不需要通道号时，通道号需下发NETDEV\_INVALID\_CHANNEL\_ID \*/

NETDEV\_GET\_AREA\_PEOPLE\_COUNT\_RULE\_INFO = 532, /\* 获取区域人数统计规则配置信息 参数详见 [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, /\* 设置区域人数统计规则配置信息 参数详见 [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, /\* 获取绊线人数统计规则配置信息 参数详见 [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, /\* 设置绊线人数统计规则配置信息 参数详见 [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, /\* 获取绊线人数统计人数清零配置 参数详见 [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, /\* 设置绊线人数统计人数清零配置 参数详见 [NETDEV\_LINE\_PEOPLE\_COUNT\_RESET\_INFO\_S](#_绊线人数统计清零信息) \*/

NETDEV\_GET\_BOX\_DOME\_LINKAGE\_INFO = 547, /\* 获取枪机上已添加的球机信息 参数详见 [NETDEV\_LINK\_DOME\_INFO\_LIST\_S](#_联动球机信息列表) \*/

NETDEV\_CFG\_INVALID = 0xFFFF /\* 无效值 \*/

}NETDEV\_CONFIG\_COMMAND\_E;

### IP协议类型

typedef enum tagHOSTType

{

NETDEV\_NETWORK\_HOSTTYPE\_IPV4 = 0, /\* IPv4 \*/

NETDEV\_NETWORK\_HOSTTYPE\_IPV6 = 1, /\* IPv6 \*/

NETDEV\_NETWORK\_HOSTTYPE\_DNS = 2, /\* 域名\*/

NETDEV\_NETWORK\_HOSTTYPE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_HOSTTYPE\_E;

### 视频图像质量

typedef enum tagNETDEVVideoQuality

{

NETDEV\_VQ\_L0 = 0, /\* 最高 Highest \*/

NETDEV\_VQ\_L1 = 1,

NETDEV\_VQ\_L2 = 4, /\* 较高 Higher \*/

NETDEV\_VQ\_L3 = 8,

NETDEV\_VQ\_L4 = 12, /\* 中 Medium \*/

NETDEV\_VQ\_L5 = 16,

NETDEV\_VQ\_L6 = 20, /\* 低 Low \*/

NETDEV\_VQ\_L7 = 24,

NETDEV\_VQ\_L8 = 28, /\* 较低 Lower \*/

NETDEV\_VQ\_L9 = 31, /\* 最低 Lowest \*/

NETDEV\_VQ\_LEVEL\_INVALID = -1 /\* 无效 Valid \*/

}NETDEV\_VIDEO\_QUALITY\_E;

### 主码流类型

typedef enum tagNETDEVMainStreamType

{

NETDEV\_MAIN\_STREAM\_TYPE\_TIME = 0, /\* 定时主码流\*/

NETDEV\_MAIN\_STREAM\_TYPE\_EVENT = 1 /\* 事件主码流\*/

}NETDEV\_MAIN\_STREAM\_TYPE\_E;

### 码率模式类型

typedef enum tagNETDEVBitRateType

{

NETDEV\_BIT\_RATE\_TYPE\_CBR = 0, /\* CBR模式\*/

NETDEV\_BIT\_RATE\_TYPE\_VBR = 1 /\* VBR模式\*/

}NETDEV\_BIT\_RATE\_TYPE\_E;

### GOP类型

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;

### 图像扩展编码模式类型

typedef enum tagNETDEVSmartEncodeMode

{

NETDEV\_SMART\_ENCODE\_MODE\_OFF = 0, /\* 关闭 \*/

NETDEV\_SMART\_ENCODE\_MODE\_BASIC = 1, /\* 基础智能编码模式 \*/

NETDEV\_SMART\_ENCODE\_MODE\_ADVANCED = 2 /\* 高级智能编码模式 \*/

}NETDEV\_SMART\_ENCODE\_MODE\_E;

### OSD时间格式能力集枚举

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类型

typedef enum tagNETDEVOSDTextType

{

NETDEV\_OSD\_TEXT\_DATAANDTIME = 0, /\* 日期.时间, 如yyyy-MM-dd.HH:mm:ss, 日期格式参见#[NETDEV\_OSD\_DATE\_FORMAT\_CAP\_E](#_OSD日期格式能力集), 时间格式参见#[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;

### OSD字体形式枚举

typedef enum tagNETDEVOSDFontStyle

{

NETDEV\_OSD\_FONT\_STYLE\_BACKGROUND = 0, /\* 背景\*/

NETDEV\_OSD\_FONT\_STYLE\_STROKE = 1, /\* 描边\*/

NETDEV\_OSD\_FONT\_STYLE\_HOLLOW = 2, /\* 空心\*/

NETDEV\_OSD\_FONT\_STYLE\_NORMAL = 3 /\* 正常\*/

}NETDEV\_OSD\_FONT\_STYLE\_E;

### OSD字体大小枚举

typedef enum tagNETDEVOSDFontSize

{

NETDEV\_OSD\_FONT\_SIZE\_LARGE = 0, /\* 特大\*/

NETDEV\_OSD\_FONT\_SIZE\_BIG = 1, /\* 大\*/

NETDEV\_OSD\_FONT\_SIZE\_MEDIUM = 2, /\* 中\*/

NETDEV\_OSD\_FONT\_SIZE\_SMALL = 3 /\* 小\*/

}NETDEV\_OSD\_FONT\_SIZE\_E;

### 日期格式

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, /\* yyyy年MM月dd日yyyy/MM/dd\*/

NETDEV\_OSD\_DATE\_FORMAT\_CHINESE\_MMDD\_YYYY = 3, /\* MM月dd日yyyy年MM/dd/yyyy\*/

NETDEV\_OSD\_DATE\_FORMAT\_CHINESE\_YYY\_MMDD\_X = 4, /\* yyyy年MM月dd日星期X。dddd yyyy MM dd\*/

NETDEV\_OSD\_DATE\_FORMAT\_CHINESE\_MMDD\_YYYY\_X = 5, /\* MM月dd日yyyy年星期X。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;

### OSD时间格式枚举

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;

### OSD区域内对齐枚举

typedef enum tagNETDEVOSDAlign

{

NETDEV\_OSD\_ALIGN\_LEFT = 0, /\* 左对齐\*/

NETDEV\_OSD\_ALIGN\_RIGHT = 1 /\* 右对齐\*/

}NETDEV\_OSD\_ALIGN\_E;

### OSD边缘空的字符大小枚举

typedef enum tagNETDEVOSDMinMargin

{

NETDEV\_OSD\_MIN\_MARGIN\_NONE = 0, /\* 无\*/

NETDEV\_OSD\_MIN\_MARGIN\_SINGLE = 1, /\* 一个字符宽度\*/

NETDEV\_OSD\_MIN\_MARGIN\_DOUBLE = 2 /\* 两个字符宽度\*/

}NETDEV\_OSD\_MIN\_MARGIN\_E;

### 输入开关量运行模式

typedef enum tagNETDEVBooleanMode

{

NETDEV\_BOOLEAN\_MODE\_OPEN = 1, /\* 常开 \*/

NETDEV\_BOOLEAN\_MODE\_CLOSE = 2, /\* 常闭 \*/

NETDEV\_BOOLEAN\_MODE\_INVALID

}NETDEV\_BOOLEAN\_MODE\_E;

### 控制手动告警开关量状态命令

typedef enum tagNETDEVOutPutManualAlarmCmd

{

NETDEV\_MANUAL\_ALARM\_CMD\_CLOSE = 0, /\* 关闭\*/

NETDEV\_MANUAL\_ALARM\_CMD\_TRIGGER = 1 /\* 触发 \*/

}NETDEV\_MANUAL\_ALARM\_CMD\_E;

### 曝光模式

typedef enum tagNETDEVExposureMode

{

NETDEV\_EXPOSURE\_MODE\_AUTOMATIC = 0, /\* 自动曝光\*/

NETDEV\_EXPOSURE\_MODE\_CUSTOM = 1, /\* 自定义曝光\*/

NETDEV\_EXPOSURE\_MODE\_SHUTTER\_PRIORITY = 2, /\* 快门优先\*/

NETDEV\_EXPOSURE\_MODE\_APERTURE\_PRIORITY = 3, /\* 光圈优先\*/

NETDEV\_EXPOSURE\_MODE\_GAIN\_PRIORITY = 4, /\* 增益优先\*/

NETDEV\_EXPOSURE\_MODE\_INDOOR\_50\_HZ = 5, /\* 室内Hz \*/

NETDEV\_EXPOSURE\_MODE\_INDOOR\_60\_HZ = 6, /\* 室内Hz \*/

NETDEV\_EXPOSURE\_MODE\_MANUAL\_OPERATION = 7, /\* 手动曝光\*/

NETDEV\_EXPOSURE\_MODE\_LOWER\_GHOSTING = 8, /\* 低拖影\*/

NETDEV\_EXPOSURE\_MODE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_EXPOSURE\_MODE\_E;

### 快门时间支持的取值

typedef enum tagNETDEVShutterTimeRange

{

NETDEV\_SHUTTER\_TIME\_AUTOMATIC = 0, /\* 快门时间为自动\*/

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 /\* 无效值\*/

}NETDEV\_SHUTTER\_TIME\_RANGE\_E;

### 测光控制模式

typedef enum tagNETDEVWideDynamicMode

{

NETDEV\_WDM\_CLOSED = 0, /\* 关闭\*/

NETDEV\_WDM\_OPEN = 1, /\* 开启\*/

NETDEV\_WDM\_AUTO = 2, /\* 自动\*/

NETDEV\_WDM\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_WIDE\_DYNAMIC\_MODE\_E;

### 昼夜模式类型

typedef enum tagNETDEVDayNightMode

{

NETDEV\_DNM\_AUTO = 0, /\* 自动模式\*/

NETDEV\_DNM\_DAY\_COLOR = 1, /\* 白天--彩色\*/

NETDEV\_DNM\_NIGHT\_BAW = 2, /\* 夜晚--黑白\*/

NETDEV\_DNM\_LINK\_COLOR\_TO\_BLACK = 3, /\* 联动彩转黑\*/

NETDEV\_DNM\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_DAY\_NIGHT\_MODE\_E;

### 补光灯类型

typedef enum tagNETDEVLampType

{

NETDEV\_LAMP\_TYPE\_WITHOUT = 0, /\* 无补光灯\*/

NETDEV\_LAMP\_TYPE\_WHITE = 1, /\* 白光灯\*/

NETDEV\_LAMP\_TYPE\_INFRARED = 2, /\* 红外灯\*/

NETDEV\_LAMP\_TYPE\_LASER\_DEVICE = 3, /\* 激光器\*/

NETDEV\_LAMP\_TYPE\_INFRARED\_SHIELD = 4, /\* 红外护罩灯\*/

NETDEV\_LAMP\_TYPE\_EXTERNAL\_HEATING\_LIGHT = 5, /\* 外接暖光灯 \*/

NETDEV\_LAMP\_TYPE\_DOUBLE\_LIGHT = 6, /\* 双光源灯(白光+红外) \*/

NETDEV\_LAMP\_TYPE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_LAMP\_TYPE\_E;

### 白平衡模式

typedef enum tagNETDEVWhiteBalance

{

NETDEV\_WBM\_AUTO = 0, /\*自动模式\*/

NETDEV\_WBM\_TRIMMING = 1, /\*微调模式\*/

NETDEV\_WBM\_INDOOR = 2, /\*室内模式\*/

NETDEV\_WBM\_OUTDOOR = 3, /\*室外模式\*/

NETDEV\_WBM\_AUTO\_MERCURY\_LAMP = 4, /\*自动水银灯模式\*/

NETDEV\_WBM\_NIGHT\_TRIMMING = 5, /\*夜间微调模式\*/

NETDEV\_WBM\_AUTO\_NA\_LAMP = 6, /\*自动钠灯模式\*/

NETDEV\_WBM\_LOCK = 7, /\*锁定模式\*/

NETDEV\_WBM\_SELF\_ADAPT\_NIGHT\_TRIMMING = 8, /\*自适应夜间微调模式\*/

NETDEV\_WBM\_AUTO\_SECOND = 9, /\*自动模式\*/

NETDEV\_WBM\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_WHITE\_BALANCE\_MODE\_E;

### 磁盘工作状态枚举

typedef enum tagNETDEVDiskWorkStatus

{

NETDEV\_DISK\_WORK\_STATUS\_EMPTY = 0, /\* 空状态\*/

NETDEV\_DISK\_WORK\_STATUS\_UNFORMAT = 1, /\* 未格式化状态\*/

NETDEV\_DISK\_WORK\_STATUS\_FORMATING = 2, /\* 格式化状态\*/

NETDEV\_DISK\_WORK\_STATUS\_RUNNING = 3, /\* 运行状态\*/

NETDEV\_DISK\_WORK\_STATUS\_HIBERNATE = 4, /\* 休眠状态\*/

NETDEV\_DISK\_WORK\_STATUS\_ABNORMAL = 5, /\* 异常状态Abnormal \*/

NETDEV\_DISK\_WORK\_STATUS\_UNKNOWN = 6, /\* 未知状态\*/

NETDEV\_DISK\_WORK\_STATUS\_INVALID /\* 无效值Invalid value \*/

}NETDEV\_DISK\_WORK\_STATUS\_E;

### 聚焦模式

typedef enum tagNETDEVFocusMode

{

NETDEV\_FOCUS\_AUTO = 1, /\* 自动聚焦\*/

NETDEV\_FOCUS\_MANUAL = 2 /\* 手动聚焦\*/

}NETDEV\_FOCUS\_MODE\_E;

### 昼夜模式

typedef enum tagNETDEVIrCutFilterMode

{

NETDEV\_IR\_CUT\_FILTER\_ON = 0, /\* 白天模式\*/

NETDEV\_IR\_CUT\_FILTER\_OFF = 1, /\* 夜晚模式\*/

NETDEV\_IR\_CUT\_FILTER\_AUTO = 2 /\* 自动模式\*/

}NETDEV\_IR\_CUT\_FILTER\_MODE\_E;

### 透雾模式枚举

typedef enum tagNETDEVDefoggingMode

{

NETDEV\_DEFOGGING\_ON = 0, /\* 开\*/

NETDEV\_DEFOGGING\_OFF=1 /\* 关\*/

}NETDEV\_DEFOGGING\_MODE\_E;

### 音频采集口枚举

typedef enum tagNETDEVAudioCollectionPortType

{

NETDEV\_AUDIO\_COLLECTION\_PORT\_TYPE\_AUDIO\_IN = 0, /\* 音频输入口\*/

NETDEV\_AUDIO\_COLLECTION\_PORT\_TYPE\_SERIAL\_IN = 1, /\* 串口\*/

NETDEV\_AUDIO\_COLLECTION\_PORT\_TYPE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_AUDIO\_COLLECTION\_PORT\_TYPE\_E;

### 音频编码格式类型枚举

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 /\* 无效值\*/

}NETDEV\_AUDIO\_ENCODE\_FORMAT\_TYPE\_E;

### 音频采样率枚举

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 /\* 无效值\*/

}NETDEV\_AUDIO\_SAMPLING\_RATE\_E;

### 音频输入模式枚举

typedef enum tagNETDEVAudioInChlMode

{

NETDEV\_AUDIO\_CHL\_MODE\_LINE = 0, /\* LINE \*/

NETDEV\_AUDIO\_CHL\_MODE\_MIC = 1, /\* MIC \*/

NETDEV\_AUDIO\_CHL\_MODE\_RS485 = 2, /\* RS485数字拾音器\*/

NETDEV\_AUDIO\_CHL\_MODE\_SPEAKER = 3, /\* SpeaKer \*/

NETDEV\_AUDIO\_CHL\_MODE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_AUDIO\_CHL\_MODE\_E;

### 夏令时偏移时间

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;

### 星期

typedef enum tagNetDEVDayInWeek

{

NETDEV\_WEEK\_SUNDAY = 0, /\* 星期日 Sunday \*/

NETDEV\_WEEK\_MONDAY = 1, /\* 星期一 Monday \*/

NETDEV\_WEEK\_TUESDAY = 2, /\* 星期二 Tuesday \*/

NETDEV\_WEEK\_WEDNESDAY = 3, /\* 星期三 Wednesday \*/

NETDEV\_WEEK\_THURSDAY = 4, /\* 星期四 Thursday \*/

NETDEV\_WEEK\_FRIDAY = 5, /\* 星期五 Friday \*/

NETDEV\_WEEK\_SATURDAY = 6, /\* 星期六 Saturday \*/

NETDEV\_WEEK\_INVALID = 0xff /\* 无效值Invalid value \*/

}NETDEV\_DAY\_IN\_WEEK\_E;

### 警前预录时间枚举

typedef enum tagNETDEVPreRecordTime

{

NETDEV\_PRE\_RECORD\_TIME\_ZERO = 0, /\* 0秒\*/

NETDEV\_PRE\_RECORD\_TIME\_FIVE = 5, /\* 5秒\*/

NETDEV\_PRE\_RECORD\_TIME\_TEN = 10, /\* 10秒\*/

NETDEV\_PRE\_RECORD\_TIME\_TWENTY = 20, /\* 20秒\*/

NETDEV\_PRE\_RECORD\_TIME\_THIRTY = 30, /\* 30秒\*/

NETDEV\_PRE\_RECORD\_TIME\_SIXTY = 60 /\* 60秒\*/

}NETDEV\_PRE\_RECORD\_TIME\_E;

### 警后预录时间枚举定义

typedef enum tagNETDEVPostRecordTime

{

NETDEV\_POST\_RECORD\_TIME\_FIVE =5, /\* 5秒\*/

NETDEV\_POST\_RECORD\_TIME\_TEN =10, /\* 10 秒\*/

NETDEV\_POST\_RECORD\_TIME\_THIRTY =30, /\* 30 秒\*/

NETDEV\_POST\_RECORD\_TIME\_SIXTY =60, /\* 60 秒\*/

NETDEV\_POST\_RECORD\_TIME\_ONE\_HUNDRED\_AND\_TWENTY =120, /\* 120 秒\*/

NETDEV\_POST\_RECORD\_TIME\_THREE\_HUNDRED =300, /\* 300秒\*/

NETDEV\_POST\_RECORD\_TIME\_SIX\_HUNDRED =600 /\* 600秒\*/

}NETDEV\_POST\_RECORD\_TIME\_E;

### 运动检测区域类型枚举

typedef enum tagNETDEVMotionDetectAreaType

{

NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_RECTANGLE = 0, /\* 矩形区域 rectangle \*/

NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_MB = 1, /\* 宏块区域 macro block \*/

NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_INVALID = 0xff /\* 无效值Invalid value \*/

}NETDEV\_MOTION\_DETECTION\_AREA\_TYPE\_E;

### 磁盘类型

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;

### 磁盘工作模式

typedef enum tagNETDEVHDDWorkMode

{

NETDEV\_HDD\_WORK\_MODE\_COMMON = 0, /\* 普通盘\*/

NETDEV\_HDD\_WORK\_MODE\_RAID = 1, /\* 阵列盘\*/

NETDEV\_HDD\_WORK\_MODE\_HOT\_BACKUP = 2, /\* 热备盘\*/

NETDEV\_HDD\_WORK\_MODE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HDD\_WORK\_MODE\_E;

### 磁盘状态

typedef enum tagNETDEVHDDStatus

{

NETDEV\_HDD\_STATUS\_NO = 0, /\* 无硬盘\*/

NETDEV\_HDD\_STATUS\_NORMAL = 1, /\* 正常\*/

NETDEV\_HDD\_STATUS\_ABNORMAL = 2, /\* 异常\*/

NETDEV\_HDD\_STATUS\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HDD\_STATUS\_E;

### 存储容器状态

typedef enum tagNETDEVStorageContainerStatus

{

NETDEV\_STORAGE\_CONTAINER\_STATUS\_NO = 0, /\* 无硬盘/空闲\*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_UNFORMATTED = 1, /\* 未格式化\*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_FORMATTING = 2, /\* 正在格式化\*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_NORMAL = 3, /\* 硬盘状态良好\*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_SLEEP = 4, /\* 休眠\*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_ABNORMAL = 5, /\* 异常\*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_SWITCH = 6, /\* 切换中\*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_UNINSTALLED = 7, /\* 已卸载\*/

NETDEV\_STORAGE\_CONTAINER\_STATUS\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_STORAGE\_CONTAINER\_STATUS\_E;

### 存储容器属性

typedef enum tagNETDEVStorageContainerProperty

{

NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_RW = 0, /\* 读写\*/

NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_R = 1, /\* 只读\*/

NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_REDUNDANT = 2, /\* 冗余\*/

NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_STORAGE\_CONTAINER\_PROPERTY\_E;

### 地址类型

typedef enum tagNETDEVAddrType

{

NETDEV\_ADDR\_TYPE\_IPV4 = 0, /\* IPV4 \*/

NETDEV\_ADDR\_TYPE\_IPV6 = 1, /\* IPV6 \*/

NETDEV\_ADDR\_TYPE\_DOMAIN = 2, /\* 域名\*/

NETDEV\_ADDR\_TYPE\_IPV4\_IPV6 = 3, /\* IPV4和IPV6都需要\*/

NETDEV\_ADDR\_TYPE\_INVALID = 0XFFFF /\* 无效值\*/

}NETDEV\_ADDR\_TYPE\_E;

### 存储容器用途

typedef enum tagNETDEVStorageContainerUsageType

{

NETDEV\_STORAGE\_CONTAINER\_USAGE\_TYPE\_RECORD\_CAPTURE = 0, /\* 录像/抓图\*/

NETDEV\_STORAGE\_CONTAINER\_USAGE\_TYPE\_BACKUP = 1, /\* 备份\*/

NETDEV\_STORAGE\_CONTAINER\_USAGE\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_STORAGE\_CONTAINER\_USAGE\_TYPE\_E;

### 硬盘健康评估状态

typedef enum tagNETDEVHDDHealthAssessmentStatus

{

NETDEV\_HDD\_HEALTH\_ASSESSMENT\_STATUS\_NORMAL = 0, /\* 健康状态良好\*/

NETDEV\_HDD\_HEALTH\_ASSESSMENT\_STATUS\_PART\_DAMAGE = 1, /\* 存在坏扇区\*/

NETDEV\_HDD\_HEALTH\_ASSESSMENT\_STATUS\_FAULT = 2, /\* 故障\*/

NETDEV\_HDD\_HEALTH\_ASSESSMENT\_STATUS\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HDD\_HEALTH\_ASSESSMENT\_STATUS\_E;

### 硬盘Smart检测状态

typedef enum tagNETDEVHDDSmartCheckStatus

{

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_NOT = 0, /\* 未检测\*/

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_IN\_PORGRESS = 1, /\* 正在自检\*/

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_SUCCESS = 2, /\* 自检成功\*/

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_RECOGNITION\_FAIL = 3, /\* 硬盘识别失败\*/

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_FAIL = 4, /\* SMART自检失败\*/

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_NOT\_SUPPORT = 5, /\* 硬盘不支持检测\*/

NETDEV\_HDD\_SMART\_CHECK\_STATUS\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HDD\_SMART\_CHECK\_STATUS\_E;

### 硬盘Smart检测类型

typedef enum tagNETDEVHDDSmartCheckType

{

NETDEV\_HDD\_SMART\_CHECK\_TYPE\_BRIEF = 0, /\* 简短型\*/

NETDEV\_HDD\_SMART\_CHECK\_TYPE\_EXTEND = 1, /\* 扩展型\*/

NETDEV\_HDD\_SMART\_CHECK\_TYPE\_TRANSMISSION = 2, /\* 传输型\*/

NETDEV\_HDD\_SMART\_CHECK\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_HDD\_SMART\_CHECK\_TYPE\_E;

### 管理服务器接入协议类型

typedef enum tagNETDEVManagerServerProtocolType

{

NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_IMOS = 0, /\* IMOS \*/

NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_GB = 1, /\* 国标\*/

NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_NONE = 2, /\* 无\*/

NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_VISS = 3, /\* VISS \*/

NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_LY = 4 /\* 立元\*/

}NETDEV\_MANAGER\_SERVER\_PROTOCOL\_TYPE\_E;

### 存储容量单位

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 /\* 无效值\*/

}NETDEV\_STORAGE\_CAP\_UNIT\_E;

### 传输协议

typedef enum tagNETDEVTransProtocol

{

NETDEV\_TRANS\_PROTOCOL\_TCP = 0, /\* TCP \*/

NETDEV\_TRANS\_PROTOCOL\_UDP = 1 /\* UDP \*/

}NETDEV\_TRANS\_PROTOCOL\_E;

### 存储资源

typedef enum tagNETDEVStorageResType

{

NETDEV\_STORAGE\_RES\_TYPE\_SD = 0, /\* SD卡\*/

NETDEV\_STORAGE\_RES\_TYPE\_NAS = 1, /\* NAS \*/

NETDEV\_STORAGE\_RES\_TYPE\_INVALID = 0xff /\* 无效值\*/

}NETDEV\_STORAGE\_RES\_TYPE\_E;

### 满存储策略

typedef enum tagNETDEVFullStrategyType

{

NETDEV\_FULL\_STRATEGY\_TYPE\_FULL\_COVERAGE = 0, /\* 满覆盖\*/

NETDEV\_FULL\_STRATEGY\_TYPE\_FULL\_STOP = 1, /\* 满即停\*/

NETDEV\_FULL\_STRATEGY\_TYPE\_INVALIDP = 0xff /\* 无效值\*/

}NETDEV\_FULL\_STRATEGY\_TYPE\_E;

### 实况业务流索引

typedef enum tagNETDEVLiveStreamIndex

{

NETDEV\_LIVE\_STREAM\_INDEX\_MAIN = 0, /\* 主流 \*/

NETDEV\_LIVE\_STREAM\_INDEX\_AUX = 1, /\* 辅流 \*/

NETDEV\_LIVE\_STREAM\_INDEX\_THIRD = 2, /\* 第三流 \*/

NETDEV\_LIVE\_STREAM\_INDEX\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_LIVE\_STREAM\_INDEX\_E;

### 图片存储模式

typedef enum tagNETDEVPicStorageMode

{

NETDEV\_PIC\_STORAGE\_MODE\_NORMAL = 0, /\* 正常，上传服务器成功后删掉SD中的文件\*/

NETDEV\_PIC\_STORAGE\_MODE\_PERMANENT = 1, /\* 永存，上传服务器不管是否成功，都会保存SD卡中的文件\*/

NETDEV\_PIC\_STORAGE\_MODE\_RETAIN = 2, /\* 保留，上传服务器失败后保留SD卡上的文件\*/

NETDEV\_PIC\_STORAGE\_MODE\_INVALID = 0xff /\* 无效值\*/

}NETDEV\_PIC\_STORAGE\_MODE\_E;

### 后焦控制操作命令

typedef enum tagNETDEVBackFocusCmd

{

NETDEV\_BACKFOCUS\_CMD\_RESET = 0, /\* 自动后焦复位Para1可选 \*/

NETDEV\_BACKFOCUS\_CMD\_AUTO = 1, /\* 表示自动后焦Para1:自动后焦时间(ms) \*/

NETDEV\_BACKFOCUS\_CMD\_MANUAL = 2, /\* 表示手动后焦Para1:0-停止, 1-MBF+, 2-MBF- \*/

NETDEV\_BACKFOCUS\_CMD\_INVALID = 0xffff/\* 无效值\*/

}NETDEV\_BACKFOCUS\_CMD\_E;

### 时区

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;

### 补光灯支持的控制模式

typedef enum tagNETDEVLampCtrlMode

{

NETDEV\_LCM\_GLOBAL\_ATTENTION\_AC = 0, /\* 全局兼顾自动控制\*/

NETDEV\_LCM\_OVEREXPOSE\_INHIBIT\_AC = 1, /\* 过曝抑制自动控制\*/

NETDEV\_LCM\_PRESET\_RCM = 2, /\* 预置控制-道路模式\*/

NETDEV\_LCM\_MANUAL\_CM = 3, /\* 手动控制模式\*/

NETDEV\_LCM\_PRESET\_PCM = 4, /\* 预置控制-园区模式\*/

NETDEV\_LCM\_PHOTOSENS\_AM = 5, /\* 光敏自动模式\*/

NETDEV\_LCM\_INDOOR\_AM = 6, /\* 室内自动模式\*/

NETDEV\_LCM\_MANUAL\_FOM = 7, /\* 手动-强制开启模式\*/

NETDEV\_LCM\_NIGHT\_VIDEO\_DET\_MODE = 8, /\* 夜间视频检测模式\*/

NETDEV\_LCM\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_LAMP\_CTRL\_MODE\_E;

### 硬盘Smart评估状态

typedef enum tagNETDEVHDDSmartAssessmentStatus

{

NETDEV\_HDD\_SMART\_ASSESSMENT\_STATUS\_NORMAL = 0, /\* 良好\*/

NETDEV\_HDD\_SMART\_ASSESSMENT\_STATUS\_WARNING = 1, /\* 警告\*/

NETDEV\_HDD\_SMART\_ASSESSMENT\_STATUS\_FAULT = 2, /\* 故障\*/

NETDEV\_HDD\_SMART\_ASSESSMENT\_STATUS\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_HDD\_SMART\_ASSESSMENT\_STATUS\_E;

### 照片服务器通信协议类型

typedef enum tagNETDEVPhotoServerProType

{

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO1 = 0, /\* TCP传输: 私有V1（对应泓鎏）\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO2 = 1, /\* TCP传输: 私有V2（对应当前tms）\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO3 = 2, /\* TCP传输: 先创\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO4 = 3, /\* FTP传输: 泉视通\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO5 = 4, /\* FTP传输: 通用FTP \*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO6 = 5, /\* FTP传输: 正直\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO7 = 6, /\* webserver: 海康协议\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO8 = 7, /\* FTP传输: 中通\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO9 = 8, /\* FTP传输: 泉视通\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO10 = 9, /\* 银江，照片信息webserver 照片数据FTP传输\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO11 = 10, /\* TCP传输:立元\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO12 = 11, /\* 翔讯，照片信息TCP 照片数据FTP传输\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO31 = 12, /\* TCP传输: V2（对应当前tms短连接）\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_PRO34 = 13, /\* HTTP传输: 松立\*/

NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_PHOTO\_SERVER\_PRO\_TYPE\_E;

### 照片服务器通信子协议类型

typedef enum tagNETDEVPhotoServerSubProType

{

NETDEV\_PHOTO\_SERVER\_SUBPRO\_TYPE\_2011 = 0, /\* 智慧物联命令\*/

NETDEV\_PHOTO\_SERVER\_SUBPRO\_TYPE\_2025 = 1, /\* 智慧物联命令\*/

NETDEV\_PHOTO\_SERVER\_SUBPRO\_TYPE\_6130 = 2, /\* 智慧物联命令\*/

NETDEV\_PHOTO\_SERVER\_SUBPRO\_TYPE\_INVALID = 0xFF /\* Invalid value \*/

}NETDEV\_PHOTO\_SERVER\_SUBPRO\_TYPE\_E;

### 智能服务器接入协议类型

typedef enum tagNETDEVDataServerProtocolType

{

NETDEV\_DATA\_SERVER\_PROTOCOL\_TYPE\_TCP = 1, /\* TCP传输: 我司V2（对应当前tms）\*/

NETDEV\_DATA\_SERVER\_PROTOCOL\_TYPE\_TFTP\_COMMOM = 4, /\* FTP传输: 通用FTP \*/

NETDEV\_DATA\_SERVER\_PROTOCOL\_TYPE\_HTTP\_GA = 51 /\* HTTP传输: 视图库\*/

}NETDEV\_DATA\_SERVER\_PROTOCOL\_TYPE\_E;

### OSD内容类型

typedef enum tagNETDEVOSDContentType

{

NETDEV\_OSD\_CONTENT\_TYPE\_NOTUSE = 0, /\* 不使用\*/

NETDEV\_OSD\_CONTENT\_TYPE\_CUSTOM = 1, /\* 自定义\*/

NETDEV\_OSD\_CONTENT\_TYPE\_DATE\_AND\_TIME = 2, /\* 时间日期\*/

NETDEV\_OSD\_CONTENT\_TYPE\_PTZ\_CONTROLLER = 3, /\* 云台控制者PTZ\*/

NETDEV\_OSD\_CONTENT\_TYPE\_PTZ\_COORDINATES = 4, /\* 云台坐标PTZ\*/

NETDEV\_OSD\_CONTENT\_TYPE\_CRUISE = 5, /\* 巡航信息\*/

NETDEV\_OSD\_CONTENT\_TYPE\_ZOOM = 6, /\* 变倍信息\*/

NETDEV\_OSD\_CONTENT\_TYPE\_PRESET = 7, /\* 预置位信息\*/

NETDEV\_OSD\_CONTENT\_TYPE\_ALARM\_INFO = 8, /\* 报警信息\*/

NETDEV\_OSD\_CONTENT\_TYPE\_ENCODE = 9, /\* 编码信息\*/

NETDEV\_OSD\_CONTENT\_TYPE\_SERIAL\_PORT = 10, /\* 串口OSD \*/

NETDEV\_OSD\_CONTENT\_TYPE\_PTZ\_ORIENTATION = 11, /\* 云台方位信息PZT \*/

NETDEV\_OSD\_CONTENT\_TYPE\_CHN\_NAME = 12, /\* 通道名称\*/

NETDEV\_OSD\_CONTENT\_TYPE\_DEBUG = 13, /\* 调试OSD \*/

NETDEV\_OSD\_CONTENT\_TYPE\_PEOPLE\_COUNTING = 14, /\* 人数统计OSD \*/

NETDEV\_OSD\_CONTENT\_TYPE\_NETWORK\_PORT = 15, /\* 网口OSD \*/

NETDEV\_OSD\_CONTENT\_TYPE\_TIME = 16, /\* 时间\*/

NETDEV\_OSD\_CONTENT\_TYPE\_DATE = 17, /\* 日期\*/

NETDEV\_OSD\_CONTENT\_TYPE\_SMART\_CONTENT = 18, /\* 超感类OSD \*/

NETDEV\_OSD\_CONTENT\_TYPE\_BATTERY = 19, /\* 电池OSD \*/

NETDEV\_OSD\_CONTENT\_TYPE\_SCROLL\_OSD = 20, /\* 滚动字幕\*/

NETDEV\_OSD\_CONTENT\_TYPE\_PICTURE\_OVERLAY = 21, /\* LOGO OSD \*/

NETDEV\_OSD\_CONTENT\_TYPE\_MOTOR\_VEHICLE\_NUM = 22, /\* 天网卡口机动车流量\*/

NETDEV\_OSD\_CONTENT\_TYPE\_NON\_MOTOR\_VEHICLE\_NUM = 23, /\* 天网卡口非机动车流量\*/

NETDEV\_OSD\_CONTENT\_TYPE\_PEOPLE\_NUM = 24, /\* 天网卡口行人流量\*/

NETDEV\_OSD\_CONTENT\_TYPE\_INFOOSD\_NUM = 25 /\* INFOOSD类型数目 \*/

}NETDEV\_OSD\_CONTENT\_TYPE\_E;

### 设备类型枚举定义

typedef enum tagNETDEVDeviceMainType

{

NETDEV\_DTYPE\_MAIN\_ENCODE = 0, /\* 编码设备\*/

NETDEV\_DTYPE\_MAIN\_DECODE = 1, /\* 解码设备\*/

NETDEV\_DTYPE\_MAIN\_VMS = 2, /\* 一体机设备\*/

NETDEV\_DTYPE\_MAIN\_DA = 3, /\* 代理设备\*/

NETDEV\_DTYPE\_MAIN\_CLOUD = 4, /\* 云端编码设备 \*/

NETDEV\_DTYPE\_MAIN\_BAYONET = 5, /\* 卡口设备\*/

NETDEV\_DTYPE\_MAIN\_ACS = 6, /\* 门禁主机设备\*/

NETDEV\_DTYPE\_MAIN\_ALARMHOST = 7, /\* 报警主机设备\*/

NETDEV\_DTYPE\_MAIN\_EXITENTRANCE = 8, /\* 出入口设备\*/

NETDEV\_DTYPE\_MAIN\_LOCK = 9, /\* 锁设备\*/

NETDEV\_DTYPE\_MAIN\_PLAYER\_MANAGER = 10, /\* 播放盒管理设备\*/

NETDEV\_DTYPE\_MAIN\_PLAYER = 11, /\* 播放盒\*/

NETDEV\_DTYPE\_MAIN\_UNKNOWN = 0XFF /\* 未知设备\*/

}NETDEV\_DEVICE\_MAIN\_TYPE\_E;

### IP地址类型

typedef enum tagNETSDKIpAddressType

{

NETDEV\_IP\_TYPE\_IPV4 = 0, /\* IPV4 \*/

NETDEV\_IP\_TYPE\_IPV6 = 1, /\* IPV6 \*/

NETDEV\_IP\_TYPE\_DOMAIN = 2, /\* 域名\*/

NETDEV\_IP\_TYPE\_IPV4\_AND\_IPV6 = 3 /\* IPV4和IPV6 \*/

}NETDEV\_IP\_ADDRESS\_TYPE\_E;

### 锁信号

typedef enum tagNETDEVLockSignal

{

NETDEV\_LOCK\_SIGNAL\_STRONG = 0, /\* 强\*/

NETDEV\_LOCK\_SIGNAL\_SECOND = 1, /\* 中\*/

NETDEV\_LOCK\_SIGNAL\_WEAK = 2, /\* 弱\*/

NETDEV\_LOCK\_SIGNAL\_NO = 3, /\* 无信号\*/

NETDEV\_LOCK\_SIGNAL\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_LOCK\_SIGNAL\_E;

### 通道类型

typedef enum tagNETDEVChlType

{

NETDEV\_CHN\_TYPE\_ENCODE = 0, /\* 编码通道, 通道状态参见

[NETDEV\_CHN\_STATUS\_E](#_编码通道状态)获取数据详见 NETDEV\_DEV\_CHN\_ENCODE\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_ALARMIN = 1, /\* 告警输入通道, 通道状态参见

NETDEV\_ALARM\_RUNMODE\_E获取数据详见 NETDEV\_DEV\_CHN\_ALARMIN\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_ALARMOUT = 2, /\* 告警输出通道, 通道状态参见

NETDEV\_ALARMOUT\_CHN\_STATUS\_E获取数据详见 NETDEV\_DEV\_CHN\_ALARMOUT\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_DECODE = 3, /\* 解码通道，获取数据详见 NETDEV\_DEV\_CHN\_DECODE\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_AUDIO = 4, /\* 音频通道，获取数据详见 NETDEV\_DEV\_CHN\_AUDIO\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_NIC = 5, /\* 网卡通道\*/

NETDEV\_CHN\_TYPE\_ALARMPOINT = 6, /\* 报警点，获取数据详见 NETDEV\_DEV\_CHN\_DA\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_DOOR = 7, /\* 门，获取数据详见 NETDEV\_DEV\_CHN\_DA\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_ADU\_ENCODE = 8, /\* ADU本地编码通道, 通道状态参见

[NETDEV\_CHN\_STATUS\_E](#_编码通道状态) 获取详细信息见#NETDEV\_DEV\_CHN\_XW\_ENCODE\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_EMERGENCY = 9, /\* 紧急铃通道，获取数据详见NETDEV\_EMERGENCY\_BELL\_INFO\_S \*/

NETDEV\_CHN\_TYPE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_CHN\_TYPE\_E;

### 编码通道状态

typedef enum tagNETDEVChnStatus

{

NETDEV\_CHN\_STATUS\_OFFLINE = 0, /\* 离线offline \*/

NETDEV\_CHN\_STATUS\_ONLINE = 1, /\* 在线online \*/

NETDEV\_CHN\_STATUS\_VIDEO\_LOSE = 2, /\* 视频丢失video lose \*/

NETDEV\_CHN\_STATUS\_INVALID = 0xFF

}NETDEV\_CHN\_STATUS\_E;

### 告警输入通道状态

typedef enum tagNETDEVAlarmRunode

{

NETDEV\_ALARM\_RUNMODE\_OPEN = 0, /\* 常开 Always open \*/

NETDEV\_ALARM\_RUNMODE\_CLOSE = 1, /\* 常闭 Always closed \*/

NETDEV\_ALARM\_RUNMODE\_INVALID = 0xFF

}NETDEV\_ALARM\_RUNMODE\_E;

### 告警输出通道状态

typedef enum tagNETDEVAlarmOutChnStatus

{

NETDEV\_ALARMOUT\_CHN\_ACTIVE = 0,

NETDEV\_ALARMOUT\_CHN\_INACTIVE = 1,

NETDEV\_ALARMOUT\_CHN\_INVALID = 0xFF

}NETDEV\_ALARMOUT\_CHN\_STATUS\_E;

### 视频输入类型枚举

typedef enum tagNETDEVVideoSrcType

{

NETDEV\_VIDEO\_SRC\_TYPE\_BNC = 0, /\* BNC输入\*/

NETDEV\_VIDEO\_SRC\_TYPE\_VGA = 1, /\* VGA输入\*/

NETDEV\_VIDEO\_SRC\_TYPE\_HDMI = 2, /\* HDMI输入\*/

NETDEV\_VIDEO\_SRC\_TYPE\_DVI\_D = 3, /\* DVI\_D输入\*/

NETDEV\_VIDEO\_SRC\_TYPE\_DVI\_I = 4, /\* DVI\_I输入\*/

NETDEV\_VIDEO\_SRC\_TYPE\_DP = 5, /\* DP输入\*/

NETDEV\_VIDEO\_SRC\_TYPE\_SDI = 6, /\* SDI输入\*/

NETDEV\_VIDEO\_SRC\_TYPE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_VIDEO\_SRC\_TYPE\_E;

### 语言类型枚举

typedef enum tagNETDEVLangType

{

NETDEV\_LANG\_TYPE\_CHINESE = 0, /\* 中文\*/

NETDEV\_LANG\_TYPE\_ENGLISH = 1, /\* 英文\*/

NETDEV\_LANG\_TYPE\_TRADITIONAL\_CHINESE = 2, /\* 繁体中文\*/

NETDEV\_LANG\_TYPE\_FRENCH = 3, /\* 法语\*/

NETDEV\_LANG\_TYPE\_GERMAN = 4, /\* 德语\*/

NETDEV\_LANG\_TYPE\_SPANISH = 5, /\* 西班牙语\*/

NETDEV\_LANG\_TYPE\_ITALIAN = 6, /\* 意大利语\*/

NETDEV\_LANG\_TYPE\_PORTUGUESE = 7, /\* 葡萄牙语\*/

NETDEV\_LANG\_TYPE\_RUSSIAN = 8, /\* 俄文\*/

NETDEV\_LANG\_TYPE\_UKRAINIAN = 9, /\* 乌克兰语\*/

NETDEV\_LANG\_TYPE\_GRUANIAN = 10, /\* 格鲁尼亚语\*/

NETDEV\_LANG\_TYPE\_DUTCH = 11, /\* 荷兰语\*/

NETDEV\_LANG\_TYPE\_DANISH = 12, /\* 丹麦语\*/

NETDEV\_LANG\_TYPE\_FINNISH = 13, /\* 芬兰语\*/

NETDEV\_LANG\_TYPE\_SWEDISH = 14, /\* 瑞典语\*/

NETDEV\_LANG\_TYPE\_NORWEGIAN = 15, /\* 挪威语\*/

NETDEV\_LANG\_TYPE\_LCELANDIC = 16, /\* 冰岛语\*/

NETDEV\_LANG\_TYPE\_POLISH = 17, /\* 波兰语\*/

NETDEV\_LANG\_TYPE\_LITHUANIAN = 18, /\* 立陶宛语\*/

NETDEV\_LANG\_TYPE\_GREEK = 19, /\* 希腊语\*/

NETDEV\_LANG\_TYPE\_CZECH = 20, /\* 捷克语\*/

NETDEV\_LANG\_TYPE\_SLOVAK = 21, /\* 斯洛伐克语\*/

NETDEV\_LANG\_TYPE\_HUNGARIAN = 22, /\* 匈牙利语\*/

NETDEV\_LANG\_TYPE\_ROMAN = 23, /\* 罗马语\*/

NETDEV\_LANG\_TYPE\_SERBIAN = 24, /\* 塞尔维亚语\*/

NETDEV\_LANG\_TYPE\_CROATIAN = 25, /\* 克罗地亚语\*/

NETDEV\_LANG\_TYPE\_SLOVENIAN = 26, /\* 斯洛文尼亚语\*/

NETDEV\_LANG\_TYPE\_BULGARIAN = 27, /\* 保加利亚语\*/

NETDEV\_LANG\_TYPE\_BELARUSIAN = 28, /\* 白俄罗斯语\*/

NETDEV\_LANG\_TYPE\_TURKISH = 29, /\* 土耳其语\*/

NETDEV\_LANG\_TYPE\_ARABIC = 30, /\* 阿拉伯语\*/

NETDEV\_LANG\_TYPE\_HEBREW = 31, /\* 希伯来语\*/

NETDEV\_LANG\_TYPE\_HINDI = 32, /\* 印度语\*/

NETDEV\_LANG\_TYPE\_BENGALI = 33, /\* 孟加拉语\*/

NETDEV\_LANG\_TYPE\_PERSIAN = 34, /\* 波斯语\*/

NETDEV\_LANG\_TYPE\_JAPANESE = 35, /\* 日语\*/

NETDEV\_LANG\_TYPE\_KOREAN = 36, /\* 韩语\*/

NETDEV\_LANG\_TYPE\_THAI = 37, /\* 泰语\*/

NETDEV\_LANG\_TYPE\_MALAYSIAN = 38, /\* 马来西亚语\*/

NETDEV\_LANG\_TYPE\_INDONESIAN = 39, /\* 印度尼西亚语\*/

NETDEV\_LANG\_TYPE\_FILIPINO = 40, /\* 菲律宾语\*/

NETDEV\_LANG\_TYPE\_LAO = 41, /\* 老挝语\*/

NETDEV\_LANG\_TYPE\_VIETNAMESE = 42, /\* 越南语\*/

NETDEV\_LANG\_TYPE\_MONGOLIAN = 43, /\* 蒙古国语\*/

NETDEV\_LANG\_TYPE\_ALBANIAN = 44, /\* 阿尔巴尼亚语\*/

NETDEV\_LANG\_TYPE\_AZERBAIJANI = 45, /\* 阿塞拜疆语\*/

NETDEV\_LANG\_TYPE\_GAELIC = 46, /\* 盖尔语\*/

NETDEV\_LANG\_TYPE\_ESTONIAN = 47, /\* 爱沙尼亚语\*/

NETDEV\_LANG\_TYPE\_BOSNIAN = 48, /\* 波斯尼亚语\*/

NETDEV\_LANG\_TYPE\_LOWLAND\_SAXON = 49, /\* 低地撒克逊语\*/

NETDEV\_LANG\_TYPE\_PERUVIAN = 50, /\* 秘鲁语\*/

NETDEV\_LANG\_TYPE\_LATVIAN = 51, /\* 拉脱维亚语\*/

NETDEV\_LANG\_TYPE\_INVALID = 0xFF /\* 无效命令\*/

}NETDEV\_LANG\_TYPE\_E;

### 客户端类型

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 /\* 无效值 \*/

}NETDEV\_CLIENT\_TYPE\_E;

### 通道状态

typedef enum tagNETDEVChannelStatus

{

NETDEV\_CHL\_STATUS\_OFFLINE = 0, /\* 通道离线\*/

NETDEV\_CHL\_STATUS\_ONLINE = 1, /\* 通道在线 \*/

NETDEV\_CHL\_STATUS\_UNBIND = 2, /\* 通道空闲 \*/

NETDEV\_CHL\_STATUS\_INVALID

}NETDEV\_CHANNEL\_STATUS\_E;

### 通道类型

typedef enum tagNETDEVChannelType

{

NETDEV\_CHL\_TYPE\_DIGITAL = 0, /\* 数字通道 \*/

NETDEV\_CHL\_TYPE\_ANALOG = 1, /\* 模拟通道 \*/

NETDEV\_CHL\_TYPE\_INVALID

}NETDEV\_CHANNEL\_TYPE\_E;

### 视频制式

typedef enum tagNETDEVVideoStandard

{

NETDEV\_VIDEO\_STANDARD\_PAL = 0, /\* PAL \*/

NETDEV\_VIDEO\_STANDARD\_NTSC = 1, /\* NTSC \*/

NETDEV\_VIDEO\_STANDARD\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_VIDEO\_STANDARD\_E;

### POE口状态类型枚举

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;

### 接入设备离线原因类型枚举

typedef enum tagNETDEVChnOffReason

{

NETDEV\_CHN\_OFF\_REASON\_CONNECTING = 0, /\* 连接中\*/

NETDEV\_CHN\_OFF\_REASON\_ONLINE = 1, /\* 在线\*/

NETDEV\_CHN\_OFF\_REASON\_WRONG\_PWD = 2, /\* 用户名密码错误\*/

NETDEV\_CHN\_OFF\_REASON\_NETWORK\_ERROR = 3, /\* 网络不通\*/

NETDEV\_CHN\_OFF\_REASON\_SET\_LIVE\_STREAM\_ERROR = 4, /\* 设置实况流失败\*/

NETDEV\_CHN\_OFF\_REASON\_START\_LIVE\_STREAM\_ERROR = 5, /\* 启动实况流失败\*/

NETDEV\_CHN\_OFF\_REASON\_STREAM\_INTERUP = 6, /\* 媒体流中断\*/

NETDEV\_CHN\_OFF\_REASON\_TIMEOUT = 7, /\* 超时\*/

NETDEV\_CHN\_OFF\_REASON\_WAIT\_REGIST = 8, /\* 国标IPC等待注册报文\*/

NETDEV\_CHN\_OFF\_REASON\_MODIFY\_LIVE\_STREAM\_ERROR = 9, /\* 国标IPC修改流传输协议\*/

NETDEV\_CHN\_OFF\_REASON\_POE\_POWER\_ERROR = 10, /\* PoE口供电异常\*/

NETDEV\_CHN\_OFF\_REASON\_BW\_RECV\_NOT\_ENOUGH = 11, /\* 带宽不足\*/

NETDEV\_CHN\_OFF\_REASON\_WEAK\_PWD\_ERROR = 12, /\* 弱密码拒绝访问\*/

NETDEV\_CHN\_OFF\_REASON\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_CHN\_OFF\_REASON\_E;

### 设备状态

typedef enum tagNETDEVDeviceStatus

{

NETDEV\_DEV\_STATUS\_OFFLINE = 0, /\* 离线\*/

NETDEV\_DEV\_STATUS\_ONLINE = 1, /\* 在线\*/

NETDEV\_DEV\_STATUS\_CONNECTING = 2, /\* 连接中\*/

NETDEV\_DEV\_STATUS\_PWD\_ERROR = 3, /\* 用户名或密码错误\*/

NETDEV\_DEV\_STATUS\_NOT\_SUPPORT = 4, /\* 设备不支持\*/

NETDEV\_DEV\_STATUS\_TIMEOUT = 5, /\* 设备连接超时\*/

NETDEV\_DEV\_STATUS\_WEAK\_PWD\_ERROR = 6, /\* 远程用户弱密码错误\*/

NETDEV\_DEV\_STATUS\_NO\_DYNAMIC\_PWD = 7, /\* 设备不支持动态密码\*/

NETDEV\_DEV\_STATUS\_INVALID = 0XFF /\* 无效值\*/

}NETDEV\_DEVICE\_STATUS\_E;

### 设备子类型

typedef enum tagNETDEVDeviceSubType

{

NETDEV\_DTYPE\_SUB\_NVR = 0, /\* NVR \*/

NETDEV\_DTYPE\_SUB\_IPC = 1, /\* IPC \*/

NETDEV\_DTYPE\_SUB\_DC\_INNER = 2, /\* 内置DC \*/

NETDEV\_DTYPE\_SUB\_DC\_EXT = 3, /\* 外置DC \*/

NETDEV\_DTYPE\_SUB\_EC = 4, /\* EC \*/

NETDEV\_DTYPE\_SUB\_VMS = 5, /\* 一体机\*/

NETDEV\_DTYPE\_SUB\_DA = 6, /\* 代理设备\*/

NETDEV\_DTYPE\_SUB\_ADU = 7, /\* ADU拼控设备\*/

NETDEV\_DTYPE\_SUB\_FISHEYE\_IPC = 8, /\* 鱼眼设备\*/

NETDEV\_DTYPE\_SUB\_DECARD = 9, /\* 解码卡设备\*/

NETDEV\_DTYPE\_SUB\_FACE\_IPC = 10, /\* 人脸识别相机\*/

NETDEV\_DTYPE\_SUB\_ALARMHOST = 11, /\* 报警主机设备\*/

NETDEV\_DTYPE\_SUB\_ACS = 12, /\* 第三方门禁设备\*/

NETDEV\_DTYPE\_SUB\_FG = 13, /\* 人脸速通门设备\*/

NETDEV\_DTYPE\_SUB\_FACE\_ACS = 14, /\* 人脸门禁设备\*/

NETDEV\_DTYPE\_SUB\_PRIVATE\_GENERAL\_ACS = 15, /\* 我司通用门禁设备\*/

NETDEV\_DTYPE\_SUB\_PRIVATE\_ALARMHOST = 16, /\* 我司报警主机设备\*/

NETDEV\_DTYPE\_SUB\_LOCK\_WIF = 17, /\* WIFI锁\*/

NETDEV\_DTYPE\_SUB\_LOCK\_NBIoT = 18, /\* NBIoT锁\*/

NETDEV\_DTYPE\_SUB\_ID\_RECOGNIZE = 19, /\* 人证识别设备\*/

NETDEV\_DTYPE\_SUB\_UNKNOW = 0xFF /\* 未知设备\*/

}NETDEV\_DEVICE\_SUB\_TYPE\_E;

### 接入协议

typedef enum tagNETDEVAccessProtocol

{

NETDEV\_ACCESS\_PROTOCOL\_ONVIF = 1, /\* onvif协议\*/

NETDEV\_ACCESS\_PROTOCOL\_PRIVATE = 2, /\* 私有协议\*/

NETDEV\_ACCESS\_PROTOCOL\_GB = 3, /\* 国标\*/

NETDEV\_ACCESS\_PROTOCOL\_RTSP = 4, /\* RTSP \*/

NETDEV\_ACCESS\_PROTOCOL\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_ACCESS\_PROTOCOL\_E;

### 接入方式

typedef enum tagNETDEVDeviceAccessMode

{

NETDEV\_DEV\_ACCESS\_MODE\_IP\_DOMAIN = 0, /\* IP、域名接入\*/

NETDEV\_DEV\_ACCESS\_MODE\_P2P = 1, /\* P2P接入\*/

NETDEV\_DEV\_ACCESS\_MODE\_UNP = 2, /\* UNP接入\*/

NETDEV\_DEV\_ACCESS\_MODE\_INVALID = 0XFF /\* 无效值\*/

}NETDEV\_DEVICE\_ACCESS\_MODE\_E;

### 通道下接入的设备类型(目前仅NVR使用)

typedef enum tagNETDEVChannelCameraType

{

NETDEV\_CHL\_CAMERA\_TYPE\_FIXED = 0, /\* 固定摄像机 \*/

NETDEV\_CHL\_CAMERA\_TYPE\_PTZ = 1, /\* 云台摄像机 \*/

NETDEV\_CHL\_CAMERA\_TYPE\_FISH\_EYE = 2, /\* 鱼眼相机（鱼眼镜头+自身矫正功能）\*/

NETDEV\_CHL\_CAMERA\_TYPE\_WIDE\_ANGLE = 3, /\* 超广角相机（包含鱼眼镜头、大视角的镜头的相机 但是不具备自身矫正功能）\*/

NETDEV\_CHL\_CAMERA\_TYPE\_INVALID = 0xff

}NETDEV\_CHANNEL\_CAMERA\_TYPE\_E;

### 告警/事件上报类型

typedef enum tagNETDEVReportType

{

NETDEV\_REPORT\_TYPE\_ALARM = 0, /\* 上报类型：告警\*/

NETDEV\_REPORT\_TYPE\_EVENT = 1, /\* 上报类型：事件\*/

NETDEV\_REPORT\_TYPE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_REPORT\_TYPE\_E;

### 告警资源类型

typedef enum tagNETDEVAlarmSrcTypeEn

{

NETDEV\_ALARM\_SRC\_LOCAL\_HARD\_DISK = 0, /\*本地硬盘\*/

NETDEV\_ALARM\_SRC\_EX\_CABINET\_1\_STORAGE\_DISK = 1, /\*拓展柜-1存储盘\*/

NETDEV\_ALARM\_SRC\_EX\_CABINET\_2\_STORAGE\_DISK = 2, /\*拓展柜-2存储盘\*/

NETDEV\_ALARM\_SRC\_ARRAY\_STORAGE\_DISK = 3, /\*阵列存储盘\*/

NETDEV\_ALARM\_SRC\_NAS\_STORAGE\_DISK = 4, /\*NAS存储盘\*/

NETDEV\_ALARM\_SRC\_SAN\_STORAGE\_DISK = 5, /\*SAN存储盘\*/

NETDEV\_ALARM\_SRC\_ESATA\_STORAGE\_DISK = 6, /\*eSATA存储盘\*/

NETDEV\_ALARM\_SRC\_SD\_STORAGE\_DISK = 7, /\*SD存储盘\*/

NETDEV\_ALARM\_SRC\_VIDEO\_CHANNEL = 8, /\*视频通道\*/

NETDEV\_ALARM\_SRC\_ALARM\_INPUT\_CHANNEL = 9, /\*报警输入通道\*/

NETDEV\_ALARM\_SRC\_SYSTEM = 10, /\*系统\*/

NETDEV\_ALARM\_SRC\_AUDIO\_CHANNEL = 11, /\*音频通道\*/

NETDEV\_ALARM\_SRC\_DECODE\_CHANNEL = 12, /\*解码通道\*/

NETDEV\_ALARM\_SRC\_ALARM\_OUTPUT\_CHANNEL = 13, /\*报警输出通道\*/

NETDEV\_ALARM\_SRC\_ACCESS\_CONTROL\_CHANNEL = 14, /\*门禁通道\*/

NETDEV\_ALARM\_SRC\_ALARM\_POINT\_CHANNEL = 15, /\*报警点通道\*/

NETDEV\_ALARM\_SRC\_EMERGENCY\_BELL\_CHANNEL = 16, /\*紧急铃通道(第三方告警)\*/

NETDEV\_ALARM\_SRC\_DEV = 17, /\*设备类\*/

NETDEV\_ALARM\_SRC\_INVALID = 0xFFFF /\* 无效值\*/

}NETDEV\_ALARM\_SRC\_TYPE\_E;

### 媒体类型枚举类型定义

typedef enum tagNetSDKMediaMode

{

NETDEV\_MEDIA\_MODE\_LOCAL = 0, /\* 本地文件播放\*/

NETDEV\_MEDIA\_MODE\_REALPLAY = 1, /\* 网络流媒体实况播放\*/

NETDEV\_MEDIA\_MODE\_PLAYBACK = 2, /\* 网络流媒体回放播放\*/

NETDEV\_MEDIA\_MODE\_DOWNLOAD = 3, /\* 网络流媒体下载\*/

NETDEV\_MEDIA\_MODE\_VOICECOM = 4, /\* 语音对讲\*/

NETDEV\_MEDIA\_MODE\_VOICESRV = 5, /\* 语音数据服务\*/

NETDEV\_MEDIA\_MODE\_M3U8PLAY = 6, /\* M3u8文件播放\*/

NETDEV\_MEDIA\_MODE\_UNKNOWN = 0xff /\* 未知模式\*/

}NETDEV\_MEDIA\_MODE\_E;

### 解码层上报运行信息的类型的枚举定义

typedef enum tagNETDEVPlayerRunInfoType

{

NETDEV\_PLAYER\_RUN\_INFO\_RECORD\_VIDEO = 1, /\*\*< 本地录像过程中上报运行信息\*/

NETDEV\_PLAYER\_RUN\_INFO\_MEDIA\_PROCESS = 2, /\*\*< 视频媒体处理过程中的上报运行信息\*/

NETDEV\_PLAYER\_RUN\_INFO\_SERIES\_SNATCH = 3, /\*\*< 连续抓拍过程中上报运行信息\*/

NETDEV\_PLAYER\_RUN\_INFO\_MEDIA\_VOICE = 4, /\*\*< 语音业务过程中上报运行信息\*/

NETDEV\_PLAYER\_RUN\_INFO\_MEDIA\_NOT\_IDENTIFY = 5, /\*\*< 码流无法识别\*/

NETDEV\_PLAYER\_RUN\_INFO\_RECV\_PACKET\_NUM = 6, /\*\*< 周期内接收到的包数\*/

NETDEV\_PLAYER\_RUN\_INFO\_RECV\_BYTE\_NUM = 7, /\*\*< 周期内接收到的字节数\*/

NETDEV\_PLAYER\_RUN\_INFO\_VIDEO\_FRAME\_NUM = 8, /\*\*< 周期内解析的视频帧数\*/

NETDEV\_PLAYER\_RUN\_INFO\_AUDIO\_FRAME\_NUM = 9, /\*\*< 周期内解析的音频帧数\*/

NETDEV\_PLAYER\_RUN\_INFO\_LOST\_PACKET\_RATIO = 10, /\*\*< 周期内丢包率统计信息（单位为.01%）\*/

NETDEV\_PLAYER\_RUN\_INFO\_MEDIA\_PLAY\_PROGRESS = 11, /\*\*< 媒体中携带的进度信息\*/

NETDEV\_PLAYER\_RUN\_INFO\_MEDIA\_PLAY\_END = 12, /\*\*< 媒体中携带的播放结束\*/

NETDEV\_PLAYER\_RUN\_INFO\_MEDIA\_ABNORMAL = 13, /\*\*< 媒体处理异常\*/

NETDEV\_PLAYER\_RUN\_INFO\_CODEC = 14, /\*\*< 编解码器\*/

NETDEV\_PLAYER\_RUN\_INFO\_STREAM = 15, /\*\*< 网络流或输入流播放\*/

NETDEV\_PLAYER\_RUN\_INFO\_PLAYBACK\_FINISH = 16, /\*\*< 回放结束\*/

NETDEV\_PLAYER\_RUN\_INFO\_SNATCH = 17, /\*\*< 截图过程中的上报运行信息\*/

NETDEV\_PLAYER\_RUN\_INFO\_INVALID = 0xff

}NETDEV\_PLAYER\_RUN\_INFO\_TYPE\_E;

### 事件动作类型枚举

typedef enum tagNETDEVEventActionType

{

NETDEV\_EVENT\_ACTION\_TYPE\_ADD = 0, /\* 事件动作类型：增加\*/

NETDEV\_EVENT\_ACTION\_TYPE\_DELETE = 1, /\* 事件动作类型：删除\*/

NETDEV\_EVENT\_ACTION\_TYPE\_MODIFY = 2, /\* 事件动作类型：修改\*/

NETDEV\_EVENT\_ACTION\_TYPE\_ONLINE = 3, /\* 事件动作类型：上线\*/

NETDEV\_EVENT\_ACTION\_TYPE\_OFFLINE = 4, /\* 事件动作类型：离线\*/

NETDEV\_EVENT\_ACTION\_TYPE\_EMAP\_ALARM = 5, /\* 事件动作类型：电子地图告警\*/

NETDEV\_EVENT\_ACTION\_TYPE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_EVENT\_ACTION\_TYPE\_E;

### 事件资源类型枚举

typedef enum tagNETDEVEventResType

{

NETDEV\_EVENT\_RES\_TYPE\_USER = 0, /\* 用户资源，用户上下线对应登录句柄，其余对应用户编号\*/

NETDEV\_EVENT\_RES\_TYPE\_DEVICE = 1, /\* 设备资源，对应设备编号\*/

NETDEV\_EVENT\_RES\_TYPE\_CHANNEL = 2, /\* 通道资源，对应通道编号\*/

NETDEV\_EVENT\_RES\_TYPE\_LOGOUT = 3, /\* 强制用户退出，对应登录句柄\*/

NETDEV\_EVENT\_RES\_TYPE\_SEQUENCE = 4, /\* 轮巡资源，对应资源ID \*/

NETDEV\_EVENT\_RES\_TYPE\_EMAP\_HOTPO = 5, /\* 电子地图热点资源，对应资源ID \*/

NETDEV\_EVENT\_RES\_TYPE\_EMAP\_HOTAREA = 6, /\* 电子地图热区资源，对应资源ID \*/

NETDEV\_EVENT\_RES\_TYPE\_EMAP\_ALARM = 7, /\* 电子地图告警资源，对应资源ID \*/

NETDEV\_EVENT\_RES\_TYPE\_TIMETEMPLATE = 8, /\* 告警预案模板，对应模板ID \*/

NETDEV\_EVENT\_RES\_TYPE\_SYSRIGH = 9, /\* 系统权限资源，对应用户登录句柄\*/

NETDEV\_EVENT\_RES\_TYPE\_DEVRIGHT = 10, /\* 设备权限资源，对应通道编号\*/

NETDEV\_EVENT\_RES\_TYPE\_ORG = 11, /\* 组织资源，对应组织编号\*/

NETDEV\_EVENT\_RES\_TYPE\_ALARM\_TASK = 12, /\* 报警任务资源，对应报警任务编号\*/

NETDEV\_EVENT\_RES\_TYPE\_SLAVE = 13, /\* 主从资源(与服务端保持一致) \*/

NETDEV\_EVENT\_RES\_TYPE\_TVWALL = 14, /\* 电视墙资源，对应电视墙ID \*/

NETDEV\_EVENT\_RES\_TYPE\_TVWALL\_SCENE = 15, /\* 电视墙场景资源，对应电视墙ID \*/

NETDEV\_EVENT\_RES\_TYPE\_WND = 16, /\* 电视墙窗口资源，对应窗口ID \*/

NETDEV\_EVENT\_RES\_TYPE\_VIRTUAL\_LED = 17, /\* 电视墙虚拟LED资源，对应虚拟LED ID \*/

NETDEV\_EVENT\_RES\_TYPE\_BROADCAST\_CHANGE = 18, /\* 广播组信息变更(值与服务端保持一致) \*/

NETDEV\_EVENT\_RES\_TYPE\_LOGIC\_ORG = 19, /\* 虚拟组织资源，对应组织编号，删除虚拟组织下通道时使用\*/

NETDEV\_EVENT\_RES\_TYPE\_USER\_ROLE = 20, /\* 用户角色资源，对应用户登录句柄\*/

NETDEV\_EVENT\_RES\_TYPE\_ROLE\_ORG = 21, /\* 角色组织展示树资源，对应用户登录句柄\*/

NETDEV\_EVENT\_RES\_TYPE\_EMAP\_PIC = 22, /\* 图片资源，对应热区编号\*/

NETDEV\_EVENT\_RES\_TYPE\_PATROL = 23, /\* 巡航资源\*/

NETDEV\_EVENT\_RES\_TYPE\_RECORD = 24, /\* 录制轨迹资源\*/

NETDEV\_EVENT\_RES\_TYPE\_ACS\_PERSON = 25, /\* 门禁人员资源，对应门禁人员ID \*/

NETDEV\_EVENT\_RES\_TYPE\_ACS\_PERSON\_CARD = 26, /\* 门禁卡资源，对应门禁人员ID \*/

NETDEV\_EVENT\_RES\_TYPE\_TVWALL\_LIST = 27, /\* 电视墙列表，权限到电视墙\*/

NETDEV\_EVENT\_RES\_TYPE\_TVWALL\_SCENE\_SWITCH = 28, /\* 电视墙场景切换\*/

NETDEV\_EVENT\_RES\_TYPE\_FACE\_LIB = 29, /\* 人脸库资源，对应人脸库ID \*/

NETDEV\_EVENT\_RES\_TYPE\_FACE\_CUSTOM = 30, /\* 人脸库自定义属性，对应属性ID \*/

NETDEV\_EVENT\_RES\_TYPE\_FACE\_MEMBER = 31, /\* 人脸成员资源，对应人脸库ID \*/

NETDEV\_EVENT\_RES\_TYPE\_FACE\_GUARD = 32, /\* 人脸布控资源，对应人脸布控ID \*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_DETECT = 33, /\* 智能检测资源，对应智能检测类型，人脸识别:0 \*/

NETDEV\_EVENT\_RES\_TYPE\_MANUAL\_STATUS = 34, /\* 手动录像状态\*/

NETDEV\_EVENT\_RES\_TYPE\_VEHICLE\_GUARD = 38, /\* 车牌布控资源，对应车牌布控ID \*/

NETDEV\_EVENT\_RES\_TYPE\_CDN\_CHANNEL = 39, /\* CDN通道资源变更，不上报对应变更信息，客户端收到事件后主动来查询通道列表\*/

NETDEV\_EVENT\_RES\_TYPE\_FACE\_MEMBER\_SORT = 40, /\*人脸成员划归资源，对应人脸库ID \*/

NETDEV\_EVENT\_RES\_TYPE\_VEHICLE\_LIB = 41, /\* 车辆库资源，对应车辆库ID \*/

NETDEV\_EVENT\_RES\_TYPE\_VEHICLE\_MEMBER\_SORT = 42, /\* 车辆成员划归资源，对应车辆库ID \*/

NETDEV\_EVENT\_RES\_TYPE\_VEHICLE\_MEMBER = 43, /\* 车辆成员资源，对应车辆成员ID \*/

NETDEV\_EVENT\_RES\_TYPE\_REMAIN\_PARK\_SPACE = 44, /\* 停车场剩余车位资源\*/

NETDEV\_EVENT\_RES\_TYPE\_ABNORMAL\_CHARGE = 45, /\* 异常收费规则资源\*/

NETDEV\_EVENT\_RES\_TYPE\_PARK\_RES\_CHANGE = 46, /\* 停车场资源变更\*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_COMMUNITY\_PERSON = 47, /\* 智慧社区人员资源\*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_COMMUNITY\_CAR = 48, /\* 智慧社区车辆资源\*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_COMMUNITY\_BUILDING = 49, /\* 智慧社区楼栋资源\*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_COMMUNITY\_UNIT = 50, /\* 智慧社区单元资源\*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_COMMUNITY\_ROOM = 51, /\* 智慧社区房间资源\*/

NETDEV\_EVENT\_RES\_TYPE\_SMART\_COMMUNITY\_ORG = 52, /\* 智慧社区组织(小区)资源\*/

NETDEV\_EVENT\_RES\_TYPE\_TEMP\_VEHICLE\_PAY\_CHANGE = 53, /\* 临时车缴费信息变更\*/

NETDEV\_EVENT\_RES\_TYPE\_THIRDPART\_PAYMENT = 54, /\* 停车场第三方支付资源\*/

NETDEV\_EVENT\_RES\_TYPE\_PAYMENT\_WAIT = 55, /\* 支付中，请等待\*/

NETDEV\_EVENT\_RES\_TYPE\_ENTRANCE\_LIST = 56, /\* 出入口列表，权限到出入口\*/

NETDEV\_EVENT\_RES\_TYPE\_VIEWPLAN\_RES = 57, /\* 视图计划，对应计划ID \*/

NETDEV\_EVENT\_RES\_TYPE\_SCENESPLAN\_RES = 58, /\* 场景间计划，对应计划ID \*/

NETDEV\_EVENT\_RES\_TYPE\_ACS\_PERMISSION = 59, /\* 权限资源, 用于授权信息变更\*/

NETDEV\_EVENT\_RES\_TYPE\_ACS\_GROUP = 60, /\* 门禁权限组资源，用于门禁权限组（组织）的变更\*/

NETDEV\_EVENT\_RES\_TYPE\_TVWALL\_AUDIO = 61, /\* 音频事件\*/

NETDEV\_EVENT\_RES\_TYPE\_PAYMENT\_RRULE = 62, /\* 手动添加收费规则资源 \*/

NETDEV\_EVENT\_RES\_TYPE\_GIS = 63, /\* 电子地图资源\*/

NETDEV\_EVENT\_RES\_TYPE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_EVENT\_RES\_TYPE\_E;

### 客流量统计模式类型

typedef enum tagNETDEVTrafficStatisticsType

{

NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_IN = 0, /\* 统计类型：进入\*/

NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_OUT = 1, /\* 统计类型：离开\*/

NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_ALL = 2, /\* 统计类型：所有\*/

NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_BUTT

}NETDEV\_TRAFFIC\_STATISTICS\_TYPE\_E;

### 客流量统计报表类型

typedef enum tagNETDEVFormType

{

NETDEV\_FORM\_TYPE\_DAY = 0, /\* 日报表\*/

NETDEV\_FORM\_TYPE\_WEEK = 1, /\* 周报表\*/

NETDEV\_FORM\_TYPE\_MONTH = 2, /\* 月报表\*/

NETDEV\_FORM\_TYPE\_YEAR = 3, /\* 年报表\*/

NETDEV\_FORM\_TYPE\_BUTT

}NETDEV\_FORM\_TYPE\_E;

### 客流量报表类型

typedef enum tagNETDEVTrafficStatisticsFormType

{

NETDEV\_TRAFFIC\_STAT\_FORM\_BY\_MINUTE = 0, /\* 按分钟\*/

NETDEV\_TRAFFIC\_STAT\_FORM\_BY\_HOUR = 1, /\* 按小时\*/

NETDEV\_TRAFFIC\_STAT\_FORM\_BY\_DAY = 2, /\* 按天\*/

NETDEV\_TRAFFIC\_STAT\_FORM\_BY\_MONTH = 3, /\* 按月\*/

NETDEV\_TRAFFIC\_STAT\_FORM\_BY\_BUTT

}NETDEV\_TRAFFIC\_STATIC\_FORM\_TYPE\_E;

### 分辨率枚举

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, /\* 自定义分辨率 \*/

NETDEV\_VIDEO\_FORMAT\_INVALID = 0xF

}NETDEV\_VIDEO\_FORMAT\_E;

### 分屏能力枚举

typedef enum tagNETDEVPaneCap

{

NETDEV\_PANE\_UNSUPPORT = 0, /\* 不支持 \*/

NETDEV\_PANE\_SAME\_WND = 1, /\* 支持单监视器相同大小&位置的窗口分屏 \*/

NETDEV\_PANE\_ANY\_WND = 2, /\* 支持任意窗口的分屏 \*/

NETDEV\_PANE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_XW\_PANE\_CAP\_E;

### 窗口漫游能力枚举

typedef enum tagNETDEVRoamingCap

{

NETDEV\_ROAMING\_UNSUPPORT = 0, /\* 不支持 \*/

NETDEV\_ROAMING\_SINGLE\_WND = 1, /\* 支持单分屏窗口漫游 \*/

NETDEV\_ROAMING\_PANE\_WND = 2, /\* 支持多分屏窗口漫游 \*/

NETDEV\_ROAMING\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_XW\_ROAMING\_CAP\_E;

### 开窗能力枚举

typedef enum tagNETDEVWndCap

{

NETDEV\_WINDOWS\_UNSUPPORT = 0, /\* 不支持 \*/

NETDEV\_WINDOWS\_SINGLE\_WND = 1, /\* 支持单监视器相同大小&位置的窗口 \*/

NETDEV\_WINDOWS\_MULTI\_WND = 2, /\* 支持跨多个监视器，满窗口开窗 \*/

NETDEV\_WINDOWS\_ANY\_WND = 3, /\* 任意开窗 \*/

NETDEV\_WINDOWS\_INVALID =0xff /\* 无效值 \*/

}NETDEV\_XW\_WND\_CAP\_E;

### 屏幕输出制式枚举

typedef enum tagNETDEVXWLayoutNum

{

NETDEV\_XW\_LAYOUT\_TYPE\_\_SELF\_ADAPT = 0, /\* 自适应分屏 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_01 = 100, /\* 1分屏 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_02 = 200, /\* 2分屏 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_03 = 300, /\* 3分屏 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_03\_1 = 301, /\* 3分屏 上2下1\*/

NETDEV\_XW\_LAYOUT\_TYPE\_04 = 400, /\* 4分屏 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_04\_1 = 401, /\* 4分屏1+1+2 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_05 = 500, /\* 5分屏1+3+1 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_05\_1 = 501, /\* 5分屏1+1+3 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_05\_2 = 502, /\* 5分屏1+3+1 顺序不同于500 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_05\_3 = 503, /\* 5分屏1+1+3 顺序不同于501 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_05\_4 = 504, /\* 5分屏3+1+1\*/

NETDEV\_XW\_LAYOUT\_TYPE\_05\_5 = 505, /\* 5分屏1+4\*/

NETDEV\_XW\_LAYOUT\_TYPE\_06 = 600, /\* 6分屏1+5 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_07 = 700, /\* 7分屏1+6 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_07\_1 = 701, /\* 7分屏3+1+3 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_07\_2 = 702, /\* 7分屏3+1+3(3均为上2下1) \*/

NETDEV\_XW\_LAYOUT\_TYPE\_08 = 800, /\* 8分屏1+7 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_09 = 900, /\* 9分屏 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10 = 1000, /\* 10分屏1+9 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10\_1 = 1001, /\* 10分屏2+8 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_13 = 1300, /\* 13分屏1+12 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_13\_1 = 1301, /\* 13分屏1+12 顺序不同1300 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_16 = 1600, /\* 16分屏 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_17 = 1700, /\* 17分屏1+16 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_25 = 2500, /\* 25分屏 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_32 = 3200, /\* 32分屏3+1+28 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_36 = 3600, /\* 36分屏 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_64 = 6400, /\* 64分屏 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10001 = 10001, /\* 自定义分屏一 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10002 = 10002, /\* 自定义分屏二 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10003 = 10003, /\* 自定义分屏三 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10004 = 10004, /\* 自定义分屏四 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10005 = 10005, /\* 自定义分屏五 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10006 = 10006, /\* 自定义分屏六 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10007 = 10007, /\* 自定义分屏七 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_10008 = 10008, /\* 自定义分屏八 \*/

NETDEV\_XW\_LAYOUT\_TYPE\_INVALID = 0xffff

}NETDEV\_XW\_LAYOUT\_NUM\_E;

### 视频端口枚举

typedef enum tagNETDEVVideoPortType

{

NETDEV\_VIDEO\_PORT\_BNC = 0, /\* 端口类型 BNC \*/

NETDEV\_VIDEO\_PORT\_VGA = 1, /\* 端口类型 VGA \*/

NETDEV\_VIDEO\_PORT\_HDMI = 2, /\* 端口类型 HDMI \*/

NETDEV\_VIDEO\_PORT\_DVI\_D = 3, /\* 端口类型 DVI\_D \*/

NETDEV\_VIDEO\_PORT\_DVI\_I = 4, /\* 端口类型 DVI\_D \*/

NETDEV\_VIDEO\_PORT\_DP = 5, /\* 端口类型 DVI\_D \*/

NETDEV\_VIDEO\_PORT\_SDI = 6, /\* 端口类型 DVI\_D \*/

NETDEV\_VIDEO\_PORT\_RJ45 = 7, /\* 端口类型 RJ45 \*/

NETDEV\_VIDEO\_PORT\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_VIDEO\_PORT\_TYPE\_E;

### 音频通道端口枚举

typedef enum tagNETDEVAudioPortType

{

NETDEV\_AUDIO\_PORT\_SINGLE = 0, /\* 单声道 \*/

NETDEV\_AUDIO\_PORT\_DOUBLE = 1, /\* 双声道 \*/

NETDEV\_AUDIO\_PORT\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_AUDIO\_PORT\_TYPE\_E;

### 串口类型枚举

typedef enum tagNETDEVXWSerialType

{

NETDEV\_XW\_SERIAL\_TYPE\_RS232 = 0, /\* 232串口 \*/

NETDEV\_XW\_SERIAL\_TYPE\_RS422 = 1, /\* 422 \*/

NETDEV\_XW\_SERIAL\_TYPE\_RS485 = 2, /\* 485 \*/

NETDEV\_XW\_SERIAL\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_XW\_SERIAL\_TYPE\_E;

### 虚拟LED类型枚举

typedef enum tagNETDEVLEDType

{

NETDEV\_LED\_TYPE\_STRING = 0 /\* 字符型 \*/

}NETDEV\_XW\_LED\_TYPE\_E;

### 虚拟LED修改可选命令字枚举

typedef enum tagNETDEVLedCmd

{

NETDEV\_LED\_ENABLE = 0x0001, /\* 分屏模式 udwEnable \*/

NETDEV\_LED\_TYPE = 0x0002, /\* 虚拟LED类型 dwType \*/

NETDEV\_LED\_AREA = 0x0004, /\* 虚拟LED的位置信息 stArea \*/

NETDEV\_LED\_BACKGROUND = 0x0008, /\* 底色 stBackground \*/

NETDEV\_LED\_FONTINFO = 0x0010, /\* 文字信息描述 stFontInfo \*/

NETDEV\_LED\_TEXT = 0x0020, /\* 文字内容 udwTestLineNum szText\*/

NETDEV\_LED\_ALL = 0x00FF /\* 全部 all\*/

} NETDEV\_XW\_LED\_CMD\_E;

### 虚拟LED字体枚举

typedef enum tagNETDEVLEDFontFormat

{

NETDEV\_LED\_FONT\_SONG = 0, /\* 宋体 \*/

NETDEV\_LED\_FONT\_FORMAT\_SIMHEI = 1, /\* 黑体 \*/

NETDEV\_LED\_FONT\_FORMAT\_KAITI = 2, /\* 楷体 \*/

NETDEV\_LED\_FONT\_FORMAT\_Arial = 3, /\* Arial \*/

NETDEV\_LED\_FONT\_FORMAT\_FANGSONG = 4, /\* 仿宋 \*/

NETDEV\_LED\_FONT\_FORMAT\_LISHU = 5, /\* 隶书 \*/

NETDEV\_LED\_FONT\_FORMAT\_YOUYUAN = 6 /\* 幼圆 \*/

}NETDEV\_XW\_LED\_FONT\_FORMAT\_E;

### 虚拟LED字体大小枚举

typedef enum tagNETDEVXWLEDFontSize

{

NETDEV\_LED\_FONT\_SIZE\_48X48 = 0, /\* 48X48点阵 \*/

NETDEV\_LED\_FONT\_SIZE\_64X64 = 1, /\* 64X64点阵 \*/

NETDEV\_LED\_FONT\_SIZE\_96X96 = 2, /\* 96X96点阵 \*/

NETDEV\_LED\_FONT\_SIZE\_128X128 = 3, /\* 128X128点阵 \*/

NETDEV\_LED\_FONT\_SIZE\_192X192 = 4, /\* 192X192点阵 \*/

NETDEV\_LED\_FONT\_SIZE\_256X256 = 5, /\* 256X256点阵 \*/

NETDEV\_LED\_FONT\_SIZE\_512X512 = 6, /\* 512X512点阵 \*/

NETDEV\_FONT\_SIZE\_SELF\_ADAPTION = 7 /\* 自适应 \*/

}NETDEV\_XW\_LED\_FONT\_SIZE\_E;

### 虚拟LED字间距枚举

typedef enum tagNETDEVXWLEDFontGap

{

NETDEV\_LED\_FONT\_GAP\_NORMAL = 0, /\* 正常间距 \*/

NETDEV\_LED\_FONT\_GAP\_ONE = 1, /\* 间隔2/4高 \*/

NETDEV\_LED\_FONT\_GAP\_TWO = 2, /\* 间隔4/4高 \*/

NETDEV\_LED\_FONT\_GAP\_THREE = 3, /\* 间隔6/4高 \*/

NETDEV\_LED\_FONT\_GAP\_FOUR = 4, /\* 间隔8/4高 \*/

NETDEV\_LED\_FONT\_GAP\_FIVE = 5, /\* 间隔1/4高 \*/

NETDEV\_LED\_FONT\_GAP\_SIX = 6, /\* 间隔3/4高 \*/

NETDEV\_LED\_FONT\_GAP\_SEVEN = 7, /\* 间隔5/4高 \*/

NETDEV\_LED\_FONT\_GAP\_EIGHT = 8, /\* 间隔7/4高 \*/

NETDEV\_LED\_FONT\_GAP\_NINE = 9, /\* 间隔10/4高 \*/

NETDEV\_LED\_FONT\_GAP\_INVALID = 0xFF /\* 无效值(自适应) Invalid value \*/

}NETDEV\_XW\_LED\_FONT\_GAP\_E;

### 文字水平对齐方式枚举

typedef enum tagNETDEVHorizontalAlignMode

{

NETDEV\_HORIZONTAL\_ALIGN\_LEFT = 0, /\* 左对齐 \*/

NETDEV\_HORIZONTAL\_ALIGN\_MIDDLE = 1, /\* 居中对齐 \*/

NETDEV\_HORIZONTAL\_ALIGN\_RIGHT = 2 /\* 右对齐\*/

}NETDEV\_XW\_HORIZONTAL\_ALIGN\_MODE\_E;

### 文字垂直对齐方式枚举

typedef enum tagNETDEVVerticalAlignMode

{

NETDEV\_VERTICAL\_ALIGN\_TOP = 0, /\* 上对齐 \*/

NETDEV\_VERTICAL\_ALIGN\_MIDDLE = 1, /\* 居中对齐 \*/

NETDEV\_VERTICAL\_ALIGN\_BOTTOM = 2 /\* 下对齐\*/

}NETDEV\_XW\_VERTICAL\_ALIGN\_MODE\_E;

### 文字滚动方式枚举

typedef enum tagNETDEVLEDScrollingMode

{

NETDEV\_LED\_SCROLL\_STATIC = 0, /\* 静止 \*/

NETDEV\_LED\_SCROLL\_RIGHT\_TO\_LEFT = 1, /\* 从右到左 \*/

NETDEV\_LED\_SCROLL\_LEFT\_TO\_RIGHT = 2, /\* 从左到右 \*/

NETDEV\_LED\_SCROLL\_BOTTOM\_TO\_TOP = 3, /\* 从下到上 \*/

NETDEV\_LED\_SCROLL\_TOP\_TO\_BOTTOM = 4 /\* 从上到下 \*/

}NETDEV\_XW\_LED\_SCROLLING\_MODE\_E;

### 窗口放大类型枚举

typedef enum tagNETDEVZoomType

{

NETDEV\_XW\_ZOOM\_TYPE\_NO = 0x00, /\* 没放大 NO enlarge \*/

NETDEV\_XW\_ZOOM\_TYPE\_SPLIT = 0x01, /\* 分屏放大 split enlarge \*/

NETDEV\_XW\_ZOOM\_TYPE\_WND = 0x02, /\* 窗口放大 window enlarge \*/

NETDEV\_XW\_ZOOM\_TYPE\_INVALID = 0xFF /\* 无效值 invalid vule \*/

}NETDEV\_XW\_ZOOM\_TYPE\_E;

### 窗口类型枚举

typedef enum tagNETDEVTVWallWndType

{

NETDEV\_WINDOWS\_TYPE\_NORMAL = 0, /\* 普通窗口 \*/

NETDEV\_WINDOWS\_TYPE\_ALARM = 1, /\* 告警窗口 \*/

NETDEV\_WINDOWS\_TYPE\_INVALID = 0xFF /\* 无效状态 \*/

}NETDEV\_TVWALL\_WND\_TPYE\_E;

### 缩放状态枚举

typedef enum tagNETDEVScaleEnabled

{

NETDEV\_SCALE\_ENABLED\_NO = 0, /\* 不缩放 \*/

NETDEV\_SCALE\_ENABLED\_IN = 1, /\* 按比例 \*/

NETDEV\_SCALE\_ENABLED\_FULL = 2, /\* 满比例 \*/

NETDEV\_SCALE\_ENABLED\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_SCALE\_ENABLED\_E;

### 分屏业务类型枚举

typedef enum tagNETDEVSessionType

{

NETDEV\_SESSION\_TYPE\_FREE = 0, /\* 空闲 \*/

NETDEV\_SESSION\_TYPE\_LIVE\_ON\_WALL = 1, /\* 实况上墙 \*/

NETDEV\_SESSION\_TYPE\_PLAYBACK\_ON\_WALL = 2, /\* 回放上墙 \*/

NETDEV\_SESSION\_TYPE\_TURNING\_ON\_WALL = 3 /\* 轮巡上墙 \*/

}NETDEV\_SESSION\_TYPE\_E;

### 屏幕放大类型枚举

typedef enum tagNETDEVXWScreenZoomdType

{

NETDEV\_XW\_SCREEN\_ZOOM\_TYPE\_PANE = 0, /\*分屏放大\*/

NETDEV\_XW\_SCREEN\_ZOOM\_TYPE\_WIN = 1, /\*窗口放大\*/

NETDEV\_XW\_SCREEN\_ZOOM\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_XW\_SCREEN\_ZOOM\_TYPE\_E;

### 屏幕放大模式枚举

typedef enum tagNETDEVXWScreenZoomdMode

{

NETDEV\_XW\_SCREEN\_ZOOM\_MODE\_OFF = 0, /\*恢复\*/

NETDEV\_XW\_SCREEN\_ZOOM\_MODE\_ON = 1, /\*放大\*/

NETDEV\_XW\_SCREEN\_ZOOM\_MODE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_XW\_SCREEN\_ZOOM\_MODE\_E;

### 媒体传输协议枚举

typedef enum tagNETDEVLapiTransProtocal

{

NETDEV\_LAPI\_TRANS\_PROTOCAL\_TCP = 0, /\* TCP \*/

NETDEV\_LAPI\_TRANS\_PROTOCAL\_UDP = 1 /\* UDP \*/

}NETDEV\_LAPI\_TRANS\_PROTOCAL\_E;

### 实况业务类型枚举

typedef enum tagNETDEVRealPlayType

{

NETDEV\_TYPE\_NORMAL = 0, /\* 默认普通实况 \*/

NETDEV\_TYPE\_ALARM = 1, /\* 告警联动实况 \*/

NETDEV\_TYPE\_LOCAL = 2 /\* 本地实况 \*/

}NETDEV\_XW\_REALPLAY\_TYPE\_E;

### 解码类型枚举

typedef enum tagNETDEVDecodeType

{

NETDEV\_DECODE\_TYPE\_CLIENT = 0, /\* 解码器端作为客户端 主动解码 \*/

NETDEV\_DECODE\_TYPE\_SERVER = 1 /\* 解码器端作为服务端 被动解码 \*/

}NETDEV\_XW\_DECODE\_TYPE\_E;

### 加密类型枚举

typedef enum tagNetDevAuthModel

{

NETDEV\_AUTH\_MODEL\_RSA\_10 = 0, /\* RSA加密 E公钥是10进制\*/

NETDEV\_AUTH\_MODEL\_PRI = 1, /\* 通用字符串加密 \*/

NETDEV\_AUTH\_MODEL\_RSA\_16 = 2 /\* RSA加密，E公钥是16进制 \*/

}NETDEV\_AUTH\_MODEL\_E;

### 文字位置枚举

typedef enum tagNETDEVContentPosition

{

NETDEV\_XW\_CONTENT\_POSITION\_TOP = 0, /\* 上 \*/

NETDEV\_XW\_CONTENT\_POSITION\_MIDDLE = 1, /\* 中 \*/

NETDEV\_XW\_CONTENT\_POSITION\_BOTTOM = 2 /\* 下\*/

}NETDEV\_XW\_CONTENT\_POSITION\_E;

### 资源管理类型枚举

typedef enum tagNETDEVXWManageType

{

NETDEV\_XW\_MANAGE\_TYPE\_OTHER = 0, /\* 其它 \*/

NETDEV\_XW\_MANAGE\_TYPE\_XWARE = 1, /\* 显控设备侧管理 \*/

NETDEV\_XW\_MANAGE\_TYPE\_INVALID = 0XFF /\* 无效值 invalid \*/

}NETDEV\_XW\_MANAGE\_TYPE\_E;

### 流类型枚举

typedef enum tagNETDEVStreamType

{

NETDEV\_STREAM\_TYPE\_RTP\_UNICAST = 0, /\* 单播 \*/

NETDEV\_STREAM\_TYPE\_RTP\_MULTICAST = 1, /\* 组播 \*/

NETDEV\_STREAM\_TYPE\_RTP\_INVALID = 0xFF

}NETDEV\_STREAM\_TYPE\_E;

### 场景修改类型枚举

typedef enum tagNETDEVXWSceneModifyType

{

NETDEV\_XW\_SCENE\_MODIFY\_TYPE\_NAME = 0, /\* 修改场景名 modify scene name \*/

NETDEV\_XW\_SCENE\_MODIFY\_TYPE\_SAVE\_AS = 1, /\* 将一个场景的信息保存到另一个场景,场景的数据为当前场景的数据 scene Saved as \*/

NETDEV\_XW\_SCENE\_MODIFY\_TYPE\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_XW\_SCENE\_MODIFY\_TYPE\_E;

### 场景业务模式枚举

typedef enum tagNETDEVSceneSessionMode

{

NETDEV\_SCENE\_SESSION\_MODE\_CLASSICS = 0, /\* 经典模式 \*/

NETDEV\_SCENE\_SESSION\_MODE\_ADAPTIVE\_IN\_SIGNAL\_SRC = 1, /\* 自适应输入信号源模式 \*/

NETDEV\_SCENE\_SESSION\_MODE\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_SCENE\_SESSION\_MODE\_E;

### 底图类型枚举

typedef enum tagNETDEVBaseType

{

NETDEV\_BASE\_TYPE\_JPEG = 1 /\* JPEG类型 \*/

}NETDEV\_XW\_BASE\_TYPE\_E;

### 轮巡状态枚举

typedef enum tagNETDEVXWSequenceResStatus

{

NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_RUNNING = 0, /\* 运行 \*/

NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_PAUSE = 1, /\* 暂停 \*/

NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_STOP = 2, /\* 停止 \*/

NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_XW\_SEQUENCE\_RES\_STATUS\_E;

### 轮巡播放控制枚举

typedef enum tagNETDEVXWSequencePlayControlCmd

{

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_STOP = 0, /\* 停止 \*/

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_START = 1, /\* 启动 \*/

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_PAUSE = 2, /\* 暂停 \*/

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_RESUME = 3, /\* 恢复 \*/

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_PAGEUP = 4, /\* 上一屏 \*/

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_PAGEDOWN = 5, /\* 下一屏 \*/

NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_XW\_SEQUENCE\_PLAY\_CTRL\_CMD\_E;

### 场景轮巡计划模式枚举

typedef enum tagNETDEVXWScenesPlanMode

{

NETDEV\_XW\_SCENES\_PLAN\_MODE\_TIMER = 0, /\* 定时切换 \*/

NETDEV\_XW\_SCENES\_PLAN\_MODE\_TIME\_TABLE = 1, /\* 按时间表切换 24小时计划表 \*/

NETDEV\_XW\_SCENES\_PLAN\_MODE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_XW\_SCENES\_PLAN\_MODE\_E;

### 编码格式枚举

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;

### 场景窗口修改可选命令字枚举

typedef enum tagNETDEVWndCommond

{

NETDEV\_WND\_PANE\_MOD = 0x0001, /\* 分屏模式 udwPaneMod \*/

NETDEV\_WND\_LAYER = 0x0002, /\* 图层 udwLayer \*/

NETDEV\_WND\_TRANSPARENCY = 0x0004, /\* 透明度 udwTransparency \*/

NETDEV\_WND\_AREA = 0x0008, /\* 位置信息 stArea \*/

NETDEV\_WND\_NAME = 0x0010, /\* 窗口名称 name \*/

NETDEV\_WND\_STATUS = 0x0020, /\* 是否锁定 \*/

NETDEV\_WND\_WND\_TYPE = 0x0040, /\* 窗口类型 \*/

NETDEV\_WND\_DC\_CHI\_ID = 0x0080, /\* DC通道ID \*/

NETDEV\_WND\_ZOOM\_TYPE = 0x0100, /\* 放大类型 \*/

NETDEV\_WND\_SPLIT\_INDEX = 0x0200, /\* 分屏序号 \*/

NETDEV\_WND\_SCHEME\_RES\_ID = 0x0400, /\* 轮巡资源 \*/

NETDEV\_WND\_SEQ\_STATUS = 0x0800, /\* 轮巡状态参见 \*/

NETDEV\_WND\_PANE\_INFO\_NUM = 0x1000, /\* 分屏数量 \*/

NETDEV\_WND\_SCALE\_STATUS = 0x2000, /\* 缩放状态 \*/

NETDEV\_WND\_ALL = 0xFFFF /\* 全部配置 all \*/

} NETDEV\_XW\_WND\_CMD\_E;

### 轮巡状态枚举

typedef enum tagNETDEVSeqStatus

{

NETDEV\_SEQ\_STATUS\_EMPTY = 0, /\* 未轮巡 \*/

NETDEV\_SEQ\_STATUS\_RUNNING = 1, /\* 正在轮巡 \*/

NETDEV\_SEQ\_STATUS\_STOP = 2, /\* 停止轮巡 \*/

NETDEV\_SEQ\_STATUS\_INVALID = 0xFF /\* 无效状态 \*/

}NETDEV\_SEQ\_STATUS\_E;

### 流业务类型枚举

typedef enum tagNETDEVVideoType

{

NETDEV\_PLAYBACK\_REMOTE = 0, /\* 远端回放 \*/

NETDEV\_REALPLAY\_REMOTE = 1, /\* 远端实况 \*/

NETDEV\_REALPLAY\_LOCAL = 2, /\* 本地实况 \*/

NETDEV\_SEQUENCE\_PANE = 3, /\* 单分屏轮巡 暂未实现\*/

NETDEV\_SEQUENCE\_WND = 4, /\* 窗口轮巡 \*/

NETDEV\_REALPLAY\_ALARM = 5 /\* 告警联动实况 \*/

}NETDEV\_XW\_VIDEO\_TYPE\_E;

### 门禁人员管理命令枚举

typedef enum tagNETDEVACSPersonCommondType

{

NETDEV\_ACS\_PERSON\_COMMOND\_TYPE\_GET = 0, /\* 获取 \*/

NETDEV\_ACS\_PERSON\_COMMOND\_TYPE\_MOD = 1, /\* 修改 \*/

NETDEV\_ACS\_PERSON\_COMMOND\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_ACS\_PERSON\_COMMOND\_TYPE\_E;

### 报警点通道控制命令枚举

typedef enum tagNETDEVDoorCtrlActionType

{

NETDEV\_DOORCTRL\_ACTION\_TYPE\_OPEN = 0, /\* 开门 \*/

NETDEV\_DOORCTRL\_ACTION\_TYPE\_CLOSE = 1, /\* 关门 \*/

NETDEV\_DOORCTRL\_ACTION\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_DOORCTRL\_ACTION\_TYPE\_E;

### 手动联动类型枚举

typedef enum tagNETDEVlinkageType

{

NETDEV\_LINKAGE\_TYPE\_CLICK\_TRACK = 0, /\* 点击跟踪 \*/

NETDEV\_LINKAGE\_TYPE\_ENLARGE\_FRAME = 1, /\* 拉框放大 \*/

NETDEV\_LINKAGE\_TYPE\_3D\_LOCATE = 2, /\* 3D定位 \*/

NETDEV\_LINKAGE\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_LINKAGE\_TYPE\_E;

### 人脸成员证件类型枚举

typedef enum tagNETDEVFaceMemberIDType

{

NETDEV\_FACE\_MEMBER\_ID\_TYPE\_ID\_CARD = 0, /\* 身份证 \*/

NETDEV\_FACE\_MEMBER\_ID\_TYPE\_IC\_CARD = 1, /\* IC卡 \*/

NETDEV\_FACE\_MEMBER\_ID\_TYPE\_PASSPORT = 2, /\* 护照 \*/

NETDEV\_FACE\_MEMBER\_ID\_TYPE\_DRIVING = 3, /\* 驾照 \*/

NETDEV\_FACE\_MEMBER\_ID\_TYPE\_OTHER = 99, /\* 其他 \*/

NETDEV\_FACE\_MEMBER\_ID\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_FACE\_MEMBER\_ID\_TYPE\_E;

### 图片类型枚举

typedef enum tagNETDEVTMSPersionImageType

{

NETDEV\_TMS\_PERSION\_IMAGE\_TYPE\_FULL\_VIEW = 1, /\* 全景图 \*/

NETDEV\_TMS\_PERSION\_IMAGE\_TYPE\_FACE = 2, /\* 人脸图 \*/

NETDEV\_TMS\_PERSION\_IMAGE\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_TMS\_PERSION\_IMAGE\_TYPE\_E;

### 图片格式枚举

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 /\* 无效值 \*/

}NETDEV\_TMS\_PERSION\_IMAGE\_FORMAT\_E;

### 人脸通行记录类型枚举

typedef enum tagNETDEVFacePassRecordType

{

NETDEV\_TYPE\_FACE\_PASS\_SNAPSHOT = 0, /\* 人脸抓拍\*/

NETDEV\_TYPE\_FACE\_PASS\_COM\_SUCCESS = 1, /\* 比对成功告警\*/

NETDEV\_TYPE\_FACE\_PASS\_COM\_FAIL = 2, /\* 比对失败告警\*/

NETDEV\_TYPE\_FACE\_PASS\_INVALID = 0xff /\* 无效值\*/

}NETDEV\_FACE\_PASS\_RECORD\_TYPE\_E;

### 年龄段枚举

typedef enum tagNETDEVAgeRange

{

NETDEV\_AGE\_RANGE\_UNKNOW = 0, /\* 未知\*/

NETDEV\_AGE\_RANGE\_CHILD = 1, /\* 儿童\*/

NETDEV\_AGE\_RANGE\_JUVENILE = 2, /\* 少年\*/

NETDEV\_AGE\_RANGE\_Youth = 3, /\* 青年\*/

NETDEV\_AGE\_RANGE\_MIDDLEAGE = 4, /\* 中年\*/

NETDEV\_AGE\_RANGE\_OLDAGE = 5, /\* 老年\*/

NETDEV\_AGE\_RANGE\_INVALID = 0xFF /\* 无效年龄段\*/

}NETDEV\_AGE\_RANGE\_E;

### 是否戴眼镜标志枚举

typedef enum tagNETDEVGlassFlag

{

NETDEV\_GLASS\_FLAG\_UNKNOW = 0, /\* 未知\*/

NETDEV\_GLASS\_FLAG\_NO = 1, /\* 不戴\*/

NETDEV\_GLASS\_FLAG\_YES = 2, /\* 戴\*/

NETDEV\_GLASS\_FLAG\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_GLASS\_FLAG\_E;

### 眼镜款式枚举

typedef enum tagNETDEVGlassesStyle

{

NETDEV\_GLASSES\_STYLE\_UNKNOW = 0, /\* 未知 \*/

NETDEV\_GLASSES\_STYLE\_GENERAL = 1, /\* 普通眼镜 \*/

NETDEV\_GLASSES\_STYLE\_SUNLIGHT = 2, /\* 太阳眼镜 \*/

NETDEV\_GLASSES\_STYLE\_OTHER = 99, /\* 其它 \*/

NETDEV\_GLASSES\_STYLE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_GLASSES\_STYLE\_E;

### 上衣长短款式枚举

typedef enum tagNETDEVSleevesLength

{

NETDEV\_SLEEVES\_LENGTH\_UNKNOW = 0, /\* 未知\*/

NETDEV\_SLEEVES\_LENGTH\_SHORT = 1, /\* 短袖\*/

NETDEV\_SLEEVES\_LENGTH\_LONG = 2, /\* 长袖\*/

NETDEV\_SLEEVES\_LENGTH\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_SLEEVES\_LENGTH\_E;

### 衣服颜色枚举

typedef enum tagNETDEVClothesColor

{

NETDEV\_CLOTHES\_COLOR\_BLACK\_E = 0, /\* 黑色\*/

NETDEV\_CLOTHES\_COLOR\_WHITE\_E = 1, /\* 白色\*/

NETDEV\_CLOTHES\_COLOR\_GRAY\_E = 2, /\* 灰色\*/

NETDEV\_CLOTHES\_COLOR\_RED\_E = 3, /\* 红色\*/

NETDEV\_CLOTHES\_COLOR\_BLUE\_E = 4, /\* 蓝色\*/

NETDEV\_CLOTHES\_COLOR\_YELLOW\_E = 5, /\* 黄色\*/

NETDEV\_CLOTHES\_COLOR\_ORANGE\_E = 6, /\* 橙色\*/

NETDEV\_CLOTHES\_COLOR\_BROWN\_E = 7, /\* 棕色\*/

NETDEV\_CLOTHES\_COLOR\_GREEN\_E = 8, /\* 绿色\*/

NETDEV\_CLOTHES\_COLOR\_PURPLE\_E = 9, /\* 紫色\*/

NETDEV\_CLOTHES\_COLOR\_CYAN\_E = 10, /\* 青色\*/

NETDEV\_CLOTHES\_COLOR\_PINK\_E = 11, /\* 粉色\*/

NETDEV\_CLOTHES\_COLOR\_TRANSPARENT\_E = 12, /\* 透明\*/

NETDEV\_CLOTHES\_COLOR\_SILVERYWHITE\_E = 13, /\* 银白\*/

NETDEV\_CLOTHES\_COLOR\_DARK\_E = 14, /\* 深色\*/

NETDEV\_CLOTHES\_COLOR\_LIGHT\_E = 15, /\* 浅色\*/

NETDEV\_CLOTHES\_COLOR\_COLOURLESS = 16, /\* 无色\*/

NETDEV\_CLOTHES\_COLOR\_YELLOWGREEN = 17, /\* 黄绿双色\*/

NETDEV\_CLOTHES\_COLOR\_GRADUALGREEN = 18, /\* 渐变绿色\*/

NETDEV\_CLOTHES\_COLOR\_OTHER\_E = 99, /\* 其他\*/

NETDEV\_CLOTHES\_COLOR\_UNKNOW\_E = 100, /\* 未知\*/

NETDEV\_CLOTHES\_COLOR\_INVALID = 0xFF /\* 无效值 Invalid value \*/

}NETDEV\_CLOTHES\_COLOR\_E;

### 下衣长短款式枚举

typedef enum tagNETDEVTrousersLength

{

NETDEV\_TROUSERS\_LENGTH\_UNKNOW = 0, /\* 未知 \*/

NETDEV\_TROUSERS\_LENGTH\_SHORT = 1, /\* 短裤 \*/

NETDEV\_TROUSERS\_LENGTH\_LONG = 2, /\* 长裤 \*/

NETDEV\_TROUSERS\_LENGTH\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_TROUSERS\_LENGTH\_E;

### 身体抓拍朝向枚举

typedef enum tagNETDEVBodyToward

{

NETDEV\_BODY\_TOWARD\_UNKNOW = 0, /\* 未知\*/

NETDEV\_BODY\_TOWARD\_POSITIVE = 1, /\* 正面\*/

NETDEV\_BODY\_TOWARD\_SIDE = 2, /\* 侧面\*/

NETDEV\_BODY\_TOWARD\_BACK = 3, /\* 背面\*/

NETDEV\_BODY\_TOWARD\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_BODY\_TOWARD\_E;

### 鞋子长短款式枚举

typedef enum tagNETDEVShoesTubeLength

{

NETDEV\_SHOES\_TUBE\_LENGTH\_UNKNOW = 0, /\* 未知 \*/

NETDEV\_SHOES\_TUBE\_LENGTH\_LONG = 1, /\* 长筒靴 \*/

NETDEV\_SHOES\_TUBE\_LENGTH\_SHORT = 2, /\* 短筒靴/普通鞋 \*/

NETDEV\_SHOES\_TUBE\_LENGTH\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_SHOES\_TUBE\_LENGTH\_E;

### 发型长短枚举

typedef enum tagNETDEVHairLength

{

NETDEV\_HAIR\_LENGTH\_UNKNOW = 0, /\* 未知\*/

NETDEV\_HAIR\_LENGTH\_LONG = 1, /\* 长发\*/

NETDEV\_HAIR\_LENGTH\_SHORT = 2, /\* 短发\*/

NETDEV\_HAIR\_LENGTH\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_HAIR\_LENGTH\_E;

### 是否携包标志枚举

typedef enum tagNETDEVBagFlag

{

NETDEV\_BAG\_FLAG\_NO = 0, /\* 未带包\*/

NETDEV\_BAG\_FLAG\_CARRY = 1, /\* 拎包\*/

NETDEV\_BAG\_FLAG\_BACK = 2, /\* 背包\*/

NETDEV\_BAG\_FLAG\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_BAG\_FLAG\_E;

### 是否戴口罩枚举

typedef enum tagNETDEVPersonMaskFlag

{

NETDEV\_PERSON\_MASK\_FLAG\_NOT\_WEAR = 1, /\* 不戴\*/

NETDEV\_PERSON\_MASK\_FLAG\_WEAR = 2, /\* 戴\*/

NETDEV\_PERSON\_MASK\_FLAG\_UNKNOW = 255, /\* 未知\*/

NETDEV\_PERSON\_MASK\_FLAG\_INVALID = 0xFFFF /\* 无效值\*/

}NETDEV\_PERSON\_MASK\_FLAG\_E;

### 上衣纹理枚举

typedef enum tagNETDEVCoatTexture

{

NETDEV\_CLOTHES\_TEXTURE\_NO\_PATTERNS = 1, /\* 无花纹\*/

NETDEV\_CLOTHES\_TEXTURE\_EXIST\_PATTERNS = 2, /\* 有花纹\*/

NETDEV\_CLOTHES\_TEXTURE\_UNKNOW = 255, /\* 未知\*/

NETDEV\_CLOTHES\_TEXTURE\_INVALID = 0xFFFF /\* 无效值\*/

}NETDEV\_CLOTHES\_TEXTURE\_E;

### 人员运动方向枚举

typedef enum tagNETDEVMoveDirection

{

NETDEV\_MOVE\_DIRECTION\_STATIC = 1, /\* 静止\*/

NETDEV\_MOVE\_DIRECTION\_UP = 2, /\* 向上\*/

NETDEV\_MOVE\_DIRECTION\_DOWN = 3, /\* 向下\*/

NETDEV\_MOVE\_DIRECTION\_LEFT = 4, /\* 向左\*/

NETDEV\_MOVE\_DIRECTION\_RIGHT = 5, /\* 向右\*/

NETDEV\_MOVE\_DIRECTION\_LEFTUP = 6, /\* 左上\*/

NETDEV\_MOVE\_DIRECTION\_LEFTDOWN = 7, /\* 左下\*/

NETDEV\_MOVE\_DIRECTION\_RIGHTUP = 8, /\* 右上\*/

NETDEV\_MOVE\_DIRECTION\_RIGHTDOWN = 9, /\* 右下\*/

NETDEV\_MOVE\_DIRECTION\_UNKNOW = 255, /\* 未知\*/

NETDEV\_MOVE\_DIRECTION\_INVALID = 0xFFFF /\* 无效值\*/

}NETDEV\_MOVE\_DIRECTION\_E;

### 结构化场景中非机动车速度类型枚举

typedef enum tagNETDEVSpeedType

{

NETDEV\_SPEED\_TYPE\_UNKNOW = 0, /\* 未知\*/

NETDEV\_SPEED\_TYPE\_STATIC = 1, /\* 静止\*/

NETDEV\_SPEED\_TYPE\_SLOW = 2, /\* 慢速\*/

NETDEV\_SPEED\_TYPE\_MEDIUM = 3, /\* 中速\*/

NETDEV\_SPEED\_TYPE\_FAST = 4, /\* 快速\*/

NETDEV\_SPEED\_TYPE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_SPEED\_TYPE\_E;

### 结构化场景中非机动车相对画面运动方向枚举

typedef enum tagNETDEVImageDirection

{

NETDEV\_IMAGE\_DIRECTION\_UNKNOW = 0, /\* 未知\*/

NETDEV\_IMAGE\_DIRECTION\_STATIC = 1, /\* 静止\*/

NETDEV\_IMAGE\_DIRECTION\_UP = 2, /\* 向上\*/

NETDEV\_IMAGE\_DIRECTION\_DOWN = 3, /\* 向下\*/

NETDEV\_IMAGE\_DIRECTION\_LEFT = 4, /\* 向左\*/

NETDEV\_IMAGE\_DIRECTION\_RIGH = 5, /\* 向右\*/

NETDEV\_IMAGE\_DIRECTION\_LEFTUP = 6, /\* 左上\*/

NETDEV\_IMAGE\_DIRECTION\_LEFTDOWN = 7, /\* 左下\*/

NETDEV\_IMAGE\_DIRECTION\_RIGHTUP = 8, /\* 右上\*/

NETDEV\_IMAGE\_DIRECTION\_RIGHTDOWN = 9, /\* 右下\*/

NETDEV\_IMAGE\_DIRECTION\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_IMAGE\_DIRECTION\_E;

### 非机动车类型枚举

typedef enum tagNETDEVNonVehType

{

NETDEV\_NON\_VEH\_TYPE\_UNKNOW = 0, /\* 未知\*/

NETDEV\_NON\_VEH\_TYPE\_BICYCLE = 1, /\* 人力自行车\*/

NETDEV\_NON\_VEH\_TYPE\_TRIYCLE = 2, /\* 三轮车\*/

NETDEV\_NON\_VEH\_TYPE\_MOTORCYCLE = 3, /\* 摩托车\*/

NETDEV\_NON\_VEH\_TYPE\_ELECTRIC\_BICYCLE = 4, /\* 电动自行车\*/

NETDEV\_NON\_VEH\_TYPE\_TWOWHEEL\_VEHICLE = 5, /\* 二轮车（摩托车/人力自行车/电动自行车) \*/

NETDEV\_NON\_VEH\_TYPE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_NON\_VEH\_TYPE\_E;

### 车辆类型枚举

typedef enum tagNETDEVVehicleType

{

NETDEV\_VEHICLE\_TYPE\_TRICYCLE\_E = 0, /\* 三轮车\*/

NETDEV\_VEHICLE\_TYPE\_MOTOR\_BUS\_E = 1, /\* 大客车\*/

NETDEV\_VEHICLE\_TYPE\_MIDDLE\_E = 2, /\* 中型车\*/

NETDEV\_VEHICLE\_TYPE\_SMALL\_E = 3, /\* 小型车\*/

NETDEV\_VEHICLE\_TYPE\_BIG\_E = 4, /\* 大型车\*/

NETDEV\_VEHICLE\_TYPE\_TWOWHEELVEH = 5, /\* 二轮车\*/

NETDEV\_VEHICLE\_TYPE\_MOTORCYCLE\_E = 6, /\* 摩托车\*/

NETDEV\_VEHICLE\_TYPE\_TRACTOR\_E = 7, /\* 拖拉机\*/

NETDEV\_VEHICLE\_TYPE\_AGRICULTURAL\_E = 8, /\* 农用货车\*/

NETDEV\_VEHICLE\_TYPE\_SEADAN = 9, /\* 轿车\*/

NETDEV\_VEHICLE\_TYPE\_SUV\_E = 10, /\* SUV \*/

NETDEV\_VEHICLE\_TYPE\_VAN\_E = 11, /\* 面包车\*/

NETDEV\_VEHICLE\_TYPE\_SMALLTRUCK\_E = 12, /\* 小货车\*/

NETDEV\_VEHICLE\_TYPE\_MEDIUMCAR\_E = 13, /\* 中巴车/中型客车\*/

NETDEV\_VEHICLE\_TYPE\_LARGEBUS\_E = 14, /\* 大客车/大型客车\*/

NETDEV\_VEHICLE\_TYPE\_LARGETRUCK\_E = 15, /\* 大货车/大型货车\*/

NETDEV\_VEHICLE\_TYPE\_PICKUPTRUCK\_E = 16, /\* 皮卡车\*/

NETDEV\_VEHICLE\_TYPE\_BUSINESSVEH\_E = 17, /\* MPV 商务车\*/

NETDEV\_VEHICLE\_TYPE\_SPORTSCAR\_E = 18, /\* 跑车\*/

NETDEV\_VEHICLE\_TYPE\_MINICAR\_E = 19, /\* 微型轿车\*/

NETDEV\_VEHICLE\_TYPE\_HATCHBACKCAR\_E = 20, /\* 两厢轿车\*/

NETDEV\_VEHICLE\_TYPE\_THREEBOX\_E = 21, /\* 三厢轿车\*/

NETDEV\_VEHICLE\_TYPE\_LIGHTBUS\_E = 22, /\* 轻型客车\*/

NETDEV\_VEHICLE\_TYPE\_MEDIUNTRUCK\_E = 23, /\* 中型货车\*/

NETDEV\_VEHICLE\_TYPE\_TRAILER\_E = 24, /\* 挂车\*/

NETDEV\_VEHICLE\_TYPE\_TANK\_E = 25, /\* 槽罐车\*/

NETDEV\_VEHICLE\_TYPE\_WATERCAR\_E = 26, /\* 洒水车\*/

NETDEV\_VEHICLE\_TYPE\_OTHER\_E = 998, /\* 其他\*/

NETDEV\_VEHICLE\_TYPE\_UNKNOW\_E = 999, /\* 未知\*/

NETDEV\_VEHICLE\_TYPE\_INDISTINGUISH\_E = 1000, /\* 不区分车辆类型\*/

NETDEV\_VEHICLE\_TYPE\_INVALID = 0xFFFF /\* 无效值 Invalid value \*/

}NETDEV\_VEHICLE\_TYPE\_E;

### 车牌颜色枚举

typedef enum tagNETDEVPlateColor

{

NETDEV\_PLATE\_COLOR\_BLACK\_E = 0, /\* 黑色\*/

NETDEV\_PLATE\_COLOR\_WHITE\_E = 1, /\* 白色\*/

NETDEV\_PLATE\_COLOR\_GRAY\_E = 2, /\* 灰色\*/

NETDEV\_PLATE\_COLOR\_RED\_E = 3, /\* 红色\*/

NETDEV\_PLATE\_COLOR\_BLUE\_E = 4, /\* 蓝色\*/

NETDEV\_PLATE\_COLOR\_YELLOW\_E = 5, /\* 黄色\*/

NETDEV\_PLATE\_COLOR\_ORANGE\_E = 6, /\* 橙色\*/

NETDEV\_PLATE\_COLOR\_BROWN\_E = 7, /\* 棕色\*/

NETDEV\_PLATE\_COLOR\_GREEN\_E = 8, /\* 绿色\*/

NETDEV\_PLATE\_COLOR\_PURPLE\_E = 9, /\* 紫色\*/

NETDEV\_PLATE\_COLOR\_CYAN\_E = 10, /\* 青色\*/

NETDEV\_PLATE\_COLOR\_PINK\_E = 11, /\* 粉色\*/

NETDEV\_PLATE\_COLOR\_TRANSPARENT\_E = 12, /\* 透明\*/

NETDEV\_PLATE\_COLOR\_SILVERYWHITE\_E = 13, /\* 银白\*/

NETDEV\_PLATE\_COLOR\_DARK\_E = 14, /\* 深色\*/

NETDEV\_PLATE\_COLOR\_LIGHT\_E = 15, /\* 浅色\*/

NETDEV\_PLATE\_COLOR\_COLOURLESS = 16, /\* 无色\*/

NETDEV\_PLATE\_COLOR\_YELLOWGREEN = 17, /\* 黄绿双色\*/

NETDEV\_PLATE\_COLOR\_GRADUALGREEN = 18, /\* 渐变绿色\*/

NETDEV\_PLATE\_COLOR\_OTHER\_E = 99, /\* 其他\*/

NETDEV\_PLATE\_COLOR\_UNKNOW\_E = 100, /\* 未知\*/

NETDEV\_PLATE\_COLOR\_INVALID = 0xFF /\* 无效值 Invalid value \*/

}NETDEV\_PLATE\_COLOR\_E;

### 车牌类型枚举

typedef enum tagNETDEVPlateType

{

NETDEV\_PLATE\_TYPE\_BIG\_CAR\_E = 0, /\* 大型汽车号牌\*/

NETDEV\_PLATE\_TYPE\_MINI\_CAR\_E = 1, /\* 小型汽车号牌\*/

NETDEV\_PLATE\_TYPE\_EMBASSY\_CAR\_E = 2, /\* 使馆汽车号牌\*/

NETDEV\_PLATE\_TYPE\_CONSULATE\_CAR\_E = 3, /\* 领馆汽车号牌\*/

NETDEV\_PLATE\_TYPE\_OVERSEAS\_CAR\_E = 4, /\* 境外汽车号牌\*/

NETDEV\_PLATE\_TYPE\_FOREIGN\_CAR\_E = 5, /\* 外籍汽车号牌\*/

NETDEV\_PLATE\_TYPE\_COMMON\_MOTORBIKE\_E = 6, /\* 普通摩托车号牌\*/

NETDEV\_PLATE\_TYPE\_HANDINESS\_MOTORBIKE\_E = 7, /\* 轻便摩托车号牌\*/

NETDEV\_PLATE\_TYPE\_EMBASSY\_MOTORBIKE\_E = 8, /\* 使馆摩托车号牌\*/

NETDEV\_PLATE\_TYPE\_CONSULATE\_MOTORBIKE\_E = 9, /\* 领馆摩托车号牌\*/

NETDEV\_PLATE\_TYPE\_OVERSEAS\_MOTORBIKE\_E = 10, /\* 境外摩托车号牌\*/

NETDEV\_PLATE\_TYPE\_FOREIGN\_MOTORBIKE\_E = 11, /\* 外籍摩托车号牌\*/

NETDEV\_PLATE\_TYPE\_LOW\_SPEED\_CAR\_E = 12, /\* 低速车号牌\*/

NETDEV\_PLATE\_TYPE\_TRACTOR\_E = 13, /\* 拖拉机号牌\*/

NETDEV\_PLATE\_TYPE\_TRAILER\_E = 14, /\* 挂车号牌\*/

NETDEV\_PLATE\_TYPE\_COACH\_CAR\_E = 15, /\* 教练汽车号牌\*/

NETDEV\_PLATE\_TYPE\_COACH\_MOTORBIKE\_E = 16, /\* 教练摩托车号牌\*/

NETDEV\_PLATE\_TYPE\_TEMPORARY\_ENTRY\_CAR\_E = 17, /\* 临时入境汽车号牌\*/

NETDEV\_PLATE\_TYPE\_TEMPORARY\_ENTRY\_MOTORBIKE\_E = 18, /\* 临时入境摩托车号牌\*/

NETDEV\_PLATE\_TYPE\_TEMPORARY\_DRIVING\_E = 19, /\* 临时行驶车号牌\*/

NETDEV\_PLATE\_TYPE\_POLICE\_CAR\_E = 20, /\* 警用汽车号牌\*/

NETDEV\_PLATE\_TYPE\_POLICE\_MOTORBIKE\_E = 21, /\* 警用摩托车号牌\*/

NETDEV\_PLATE\_TYPE\_AGRICULTURAL\_E = 22, /\* 原农机号牌\*/

NETDEV\_PLATE\_TYPE\_HONGKONG\_ENTRY\_EXIT\_E = 23, /\* 香港入出境号牌\*/

NETDEV\_PLATE\_TYPE\_MACAO\_ENTRY\_EXIT\_E = 24, /\* 澳门入出境号牌\*/

NETDEV\_PLATE\_TYPE\_ARMED\_POLICE\_E = 25, /\* 武警号牌\*/

NETDEV\_PLATE\_TYPE\_ARMY\_E = 26, /\* 军队号牌\*/

NETDEV\_PLATE\_TYPE\_OTHER\_E = 99, /\* 其他号牌\*/

NETDEV\_PLATE\_TYPE\_INVALID = 0xFF /\* 无效值 Invalid value \*/

}NETDEV\_PLATE\_TYPE\_E;

### 通知类型枚举

typedef enum tagNETDEVNotificationType

{

NETDEV\_NOTIFICATION\_TYPE\_REALTIME = 0, /\* 实时通知 \*/

NETDEV\_NOTIFICATION\_TYPE\_HISTORY = 1, /\* 历史通知 \*/

NETDEV\_NOTIFICATION\_TYPE\_EARLYWARN = 2, /\* 预警通知 \*/

NETDEV\_NOTIFICATION\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_NOTIFICATION\_TYPE\_E;

### 智能告警类型枚举

typedef enum tagNETDEVSmartAlarmType

{

NETDEV\_SMART\_ALARM\_TYPE\_FACE\_SNAP = 0, /\* 人脸识别抓图 \*/

NETDEV\_SMART\_ALARM\_TYPE\_VEHICLE\_SNAP = 1, /\* 车牌识别抓图 \*/

NETDEV\_SMART\_ALARM\_TYPE\_VIDEO\_STRUCT\_SNAP = 3, /\* 视频结构化抓图 \*/

NETDEV\_SMART\_ALARM\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_SMART\_ALARM\_TYPE\_E;

### LAPI协议订阅类型枚举

typedef enum tagNETDEVLAPISubType

{

NETDEV\_LAPI\_SUB\_TYPE\_PARKRECOGNIZE = 2048, /\* 停车场识别抓拍 \*/

NETDEV\_LAPI\_SUB\_TYPE\_INVALID = 0xffffffff /\* 无效值 Invalid value \*/

}NETDEV\_LAPI\_SUB\_TYPE\_E;

### 车道类型枚举

typedef enum tagNETDEVVehLaneType

{

NETDEV\_VEH\_LANE\_TYPE\_ENTRANCE = 0, /\* 入口 \*/

NETDEV\_VEH\_LANE\_TYPE\_EXIT\_NOCHARGE = 1, /\* 出口，不收费 \*/

NETDEV\_VEH\_LANE\_TYPE\_EXITE\_CHARGE = 2, /\* 出口，收费 \*/

NETDEV\_VEH\_LANE\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_VEH\_LANE\_TYPE\_E;

### 车辆放行类型枚举

typedef enum tagNETDEVVehReleaseType

{

NETDEV\_VEH\_RELEASE\_TYPE\_AUTO = 0, /\* 自动放行 \*/

NETDEV\_VEH\_RELEASE\_TYPE\_MANUAL = 1, /\* 手动放行 \*/

NETDEV\_VEH\_RELEASE\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_VEH\_RELEASE\_TYPE\_E;

### 订单支付状态枚举

typedef enum tagNETDEVOrderStatus

{

NETDEV\_ORDER\_STATUS\_FINISH = 0, /\* 未支付 \*/

NETDEV\_ORDER\_STATUS\_UNFINISH = 1, /\* 已支付 \*/

NETDEV\_ORDER\_STATUS\_PREPAYMENT = 2, /\* 预支付 \*/

NETDEV\_ORDER\_STATUS\_INVALID = 0xff /\* 无效值 Invalid value \*/

}NETDEV\_ORDER\_STATUS\_E;

### 支付类型枚举

typedef enum tagNETDEVPayType

{

NETDEV\_PAY\_TYPE\_CASH = 0, /\* 现金支付 \*/

NETDEV\_PAY\_TYPE\_ALIPAY = 1, /\* 支付宝支付 \*/

NETDEV\_PAY\_TYPE\_WECHAT = 2, /\* 微信支付 \*/

NETDEV\_PAY\_TYPE\_INVALID = 0xff /\* 无效值 Invalid value \*/

}NETDEV\_PAY\_TYPE\_E;

### 人员类型枚举

typedef enum tagNETDEVACSPersonType

{

NETDEV\_ACS\_PERSON\_TYPE\_STAFF = 0, /\* 员工 \*/

NETDEV\_ACS\_PERSON\_TYPE\_VISITOR = 1, /\* 访客 \*/

NETDEV\_ACS\_PERSON\_TYPE\_STRANGER = 2, /\* 陌生人 \*/

NETDEV\_ACS\_PERSON\_TYPE\_BLACKLIST = 3, /\* 黑名单人员 \*/

NETDEV\_ACS\_PERSON\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_ACS\_PERSON\_TYPE\_E;

### 访客状态枚举

typedef enum tagNETDEVACSVisitStaus

{

NETDEV\_ACS\_VISIT\_STATUS\_SCHEDULE = 0, /\* 预约 \*/

NETDEV\_ACS\_VISIT\_STATUS\_VISITING = 1, /\* 在访 \*/

NETDEV\_ACS\_VISIT\_STATUS\_LEAVE = 2, /\* 离访 \*/

NETDEV\_ACS\_VISIT\_STATUS\_SCHEDULE\_CANCEL = 3, /\* 预约取消 \*/

NETDEV\_ACS\_VISIT\_STATUS\_TIMEOUT = 4, /\* 超时 \*/

NETDEV\_ACS\_VISIT\_STATUS\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_ACS\_VISIT\_STATUS\_E;

### 采集来源枚举

typedef enum tagNETDEVCapSrc

{

NETDEV\_CAP\_SRC\_FACE = 1, /\* 人脸识别终端采集的人脸信息 \*/

NETDEV\_CAP\_SRC\_ENTRANCE\_GUARDCARD = 2, /\* 读卡器采集的门禁卡信息 \*/

NETDEV\_CAP\_SRC\_ID = 3, /\* 读卡器采集的身份证信息 \*/

NETDEV\_CAP\_SRC\_GATE = 4, /\* 闸机采集的闸机信息 \*/

NETDEV\_CAP\_SRC\_INVALID = 0xff /\* 无效值 Invalid value \*/

}NETDEV\_CAP\_SRC\_E;

### 匹配状态枚举

typedef enum tagNETDEVMatchStatus

{

NETDEV\_MATCH\_STATUS\_SUCCESS = 1, /\* 核验成功 \*/

NETDEV\_MATCH\_STATUS\_FAIL = 2, /\* 核验失败（比对失败) \*/

NETDEV\_MATCH\_STATUS\_NO\_MONITOR\_TIME = 3, /\* 核验失败（对比成功，不在布控时间）\*/

NETDEV\_MATCH\_STATUS\_ATTR\_ABNORMAL = 10, /\* 核验失败（对比成功，属性异常) \*/

NETDEV\_MATCH\_STATUS\_BASE\_MAP\_COLLECT\_SUCC = 41, /\* 底图采集成功 \*/

NETDEV\_MATCH\_STATUS\_BASE\_MAP\_COLLECT\_FAIL = 42, /\* 底图采集失败 \*/ NETDEV\_MATCH\_STATUS\_INVALID = 0xff /\* 无效值 Invalid value \*/

}NETDEV\_MATCH\_STATUS\_E;

### 图像格式枚举

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 /\* 无效值 \*/

}NETDEV\_IMAGE\_FORMAT\_E;

### 车牌颜色

typedef enum tagNETDEVPlateColor

{

NETDEV\_PLATE\_COLOR\_BLACK\_E = 0, /\* 黑色 \*/

NETDEV\_PLATE\_COLOR\_WHITE\_E = 1, /\* 白色 \*/

NETDEV\_PLATE\_COLOR\_GRAY\_E = 2, /\* 灰色 \*/

NETDEV\_PLATE\_COLOR\_RED\_E = 3, /\* 红色 \*/

NETDEV\_PLATE\_COLOR\_BLUE\_E = 4, /\* 蓝色 \*/

NETDEV\_PLATE\_COLOR\_YELLOW\_E = 5, /\* 黄色 \*/

NETDEV\_PLATE\_COLOR\_ORANGE\_E = 6, /\* 橙色 \*/

NETDEV\_PLATE\_COLOR\_BROWN\_E = 7, /\* 棕色 \*/

NETDEV\_PLATE\_COLOR\_GREEN\_E = 8, /\* 绿色 \*/

NETDEV\_PLATE\_COLOR\_PURPLE\_E = 9, /\* 紫色 \*/

NETDEV\_PLATE\_COLOR\_CYAN\_E = 10, /\* 青色 \*/

NETDEV\_PLATE\_COLOR\_PINK\_E = 11, /\* 粉色 \*/

NETDEV\_PLATE\_COLOR\_TRANSPARENT\_E = 12, /\* 透明 \*/

NETDEV\_PLATE\_COLOR\_SILVERYWHITE\_E = 13, /\* 银白 \*/

NETDEV\_PLATE\_COLOR\_DARK\_E = 14, /\* 深色 \*/

NETDEV\_PLATE\_COLOR\_LIGHT\_E = 15, /\* 浅色 \*/

NETDEV\_PLATE\_COLOR\_COLOURLESS = 16, /\* 无色 \*/

NETDEV\_PLATE\_COLOR\_YELLOWGREEN = 17, /\* 黄绿双色 \*/

NETDEV\_PLATE\_COLOR\_GRADUALGREEN = 18, /\* 渐变绿色 \*/

NETDEV\_PLATE\_COLOR\_OTHER\_E = 99, /\* 其他 \*/

NETDEV\_PLATE\_COLOR\_UNKNOW\_E = 100, /\* 未知 \*/

NETDEV\_PLATE\_COLOR\_INVALID = 0xFF /\* 无效值 Invalid value \*/

}NETDEV\_PLATE\_COLOR\_E;

### 车牌类型

typedef enum tagNETDEVPlateType

{

NETDEV\_PLATE\_TYPE\_BIG\_CAR\_E = 0, /\* 大型汽车号牌 \*/

NETDEV\_PLATE\_TYPE\_MINI\_CAR\_E = 1, /\* 小型汽车号牌 \*/

NETDEV\_PLATE\_TYPE\_EMBASSY\_CAR\_E = 2, /\* 使馆汽车号牌 \*/

NETDEV\_PLATE\_TYPE\_CONSULATE\_CAR\_E = 3, /\* 领馆汽车号牌 \*/

NETDEV\_PLATE\_TYPE\_OVERSEAS\_CAR\_E = 4, /\* 境外汽车号牌 \*/

NETDEV\_PLATE\_TYPE\_FOREIGN\_CAR\_E = 5, /\* 外籍汽车号牌 \*/

NETDEV\_PLATE\_TYPE\_COMMON\_MOTORBIKE\_E = 6, /\* 普通摩托车号牌 \*/

NETDEV\_PLATE\_TYPE\_HANDINESS\_MOTORBIKE\_E = 7, /\* 轻便摩托车号牌 \*/

NETDEV\_PLATE\_TYPE\_EMBASSY\_MOTORBIKE\_E = 8, /\* 使馆摩托车号牌 \*/

NETDEV\_PLATE\_TYPE\_CONSULATE\_MOTORBIKE\_E = 9, /\* 领馆摩托车号牌 \*/

NETDEV\_PLATE\_TYPE\_OVERSEAS\_MOTORBIKE\_E = 10, /\* 境外摩托车号牌 \*/

NETDEV\_PLATE\_TYPE\_FOREIGN\_MOTORBIKE\_E = 11, /\* 外籍摩托车号牌 \*/

NETDEV\_PLATE\_TYPE\_LOW\_SPEED\_CAR\_E = 12, /\* 低速车号牌 \*/

NETDEV\_PLATE\_TYPE\_TRACTOR\_E = 13, /\* 拖拉机号牌 \*/

NETDEV\_PLATE\_TYPE\_TRAILER\_E = 14, /\* 挂车号牌 \*/

NETDEV\_PLATE\_TYPE\_COACH\_CAR\_E = 15, /\* 教练汽车号牌 \*/

NETDEV\_PLATE\_TYPE\_COACH\_MOTORBIKE\_E = 16, /\* 教练摩托车号牌 \*/

NETDEV\_PLATE\_TYPE\_TEMPORARY\_ENTRY\_CAR\_E = 17, /\* 临时入境汽车号牌 \*/

NETDEV\_PLATE\_TYPE\_TEMPORARY\_ENTRY\_MOTORBIKE\_E = 18, /\* 临时入境摩托车号牌 \*/

NETDEV\_PLATE\_TYPE\_TEMPORARY\_DRIVING\_E = 19, /\* 临时行驶车号牌 \*/

NETDEV\_PLATE\_TYPE\_POLICE\_CAR\_E = 20, /\* 警用汽车号牌 \*/

NETDEV\_PLATE\_TYPE\_POLICE\_MOTORBIKE\_E = 21, /\* 警用摩托车号牌 \*/

NETDEV\_PLATE\_TYPE\_AGRICULTURAL\_E = 22, /\* 原农机号牌 \*/

NETDEV\_PLATE\_TYPE\_HONGKONG\_ENTRY\_EXIT\_E = 23, /\* 香港入出境号牌 \*/

NETDEV\_PLATE\_TYPE\_MACAO\_ENTRY\_EXIT\_E = 24, /\* 澳门入出境号牌 \*/

NETDEV\_PLATE\_TYPE\_ARMED\_POLICE\_E = 25, /\* 武警号牌 \*/

NETDEV\_PLATE\_TYPE\_ARMY\_E = 26, /\* 军队号牌 \*/

NETDEV\_PLATE\_TYPE\_OTHER\_E = 99, /\* 其他号牌 \*/

NETDEV\_PLATE\_TYPE\_INVALID = 0xFF /\* 无效值 Invalid value \*/

}NETDEV\_PLATE\_TYPE\_E;

### 车辆类型

typedef enum tagNETDEVVehicleType

{

NETDEV\_VEHICLE\_TYPE\_TRICYCLE\_E = 0, /\* 三轮车 \*/

NETDEV\_VEHICLE\_TYPE\_MOTOR\_BUS\_E = 1, /\* 大客车 \*/

NETDEV\_VEHICLE\_TYPE\_MIDDLE\_E = 2, /\* 中型车 \*/

NETDEV\_VEHICLE\_TYPE\_SMALL\_E = 3, /\* 小型车 \*/

NETDEV\_VEHICLE\_TYPE\_BIG\_E = 4, /\* 大型车 \*/

NETDEV\_VEHICLE\_TYPE\_TWOWHEELVEH = 5, /\* 二轮车 \*/

NETDEV\_VEHICLE\_TYPE\_MOTORCYCLE\_E = 6, /\* 摩托车 \*/

NETDEV\_VEHICLE\_TYPE\_TRACTOR\_E = 7, /\* 拖拉机 \*/

NETDEV\_VEHICLE\_TYPE\_AGRICULTURAL\_E = 8, /\* 农用货车 \*/

NETDEV\_VEHICLE\_TYPE\_SEADAN = 9, /\* 轿车 \*/

NETDEV\_VEHICLE\_TYPE\_SUV\_E = 10, /\* SUV \*/

NETDEV\_VEHICLE\_TYPE\_VAN\_E = 11, /\* 面包车 \*/

NETDEV\_VEHICLE\_TYPE\_SMALLTRUCK\_E = 12, /\* 小货车 \*/

NETDEV\_VEHICLE\_TYPE\_MEDIUMCAR\_E = 13, /\* 中巴车/中型客车 \*/

NETDEV\_VEHICLE\_TYPE\_LARGEBUS\_E = 14, /\* 大客车/大型客车 \*/

NETDEV\_VEHICLE\_TYPE\_LARGETRUCK\_E = 15, /\* 大货车/大型货车 \*/

NETDEV\_VEHICLE\_TYPE\_PICKUPTRUCK\_E = 16, /\* 皮卡车 \*/

NETDEV\_VEHICLE\_TYPE\_BUSINESSVEH\_E = 17, /\* MPV 商务车 \*/

NETDEV\_VEHICLE\_TYPE\_SPORTSCAR\_E = 18, /\* 跑车 \*/

NETDEV\_VEHICLE\_TYPE\_MINICAR\_E = 19, /\* 微型轿车 \*/

NETDEV\_VEHICLE\_TYPE\_HATCHBACKCAR\_E = 20, /\* 两厢轿车 \*/

NETDEV\_VEHICLE\_TYPE\_THREEBOX\_E = 21, /\* 三厢轿车 \*/

NETDEV\_VEHICLE\_TYPE\_LIGHTBUS\_E = 22, /\* 轻型客车 \*/

NETDEV\_VEHICLE\_TYPE\_MEDIUNTRUCK\_E = 23, /\* 中型货车 \*/

NETDEV\_VEHICLE\_TYPE\_TRAILER\_E = 24, /\* 挂车 \*/

NETDEV\_VEHICLE\_TYPE\_TANK\_E = 25, /\* 槽罐车 \*/

NETDEV\_VEHICLE\_TYPE\_WATERCAR\_E = 26, /\* 洒水车 \*/

NETDEV\_VEHICLE\_TYPE\_OTHER\_E = 998, /\* 其他 \*/

NETDEV\_VEHICLE\_TYPE\_UNKNOW\_E = 999, /\* 未知 \*/

NETDEV\_VEHICLE\_TYPE\_INDISTINGUISH\_E = 1000, /\* 不区分车辆类型 \*/

NETDEV\_VEHICLE\_TYPE\_INVALID = 0xFFFF /\* 无效值 Invalid value \*/

}NETDEV\_VEHICLE\_TYPE\_E;

### 结构化场景中非机动车速度类型枚举

typedef enum tagNETDEVSpeedType

{

NETDEV\_SPEED\_TYPE\_UNKNOW = 0, /\* 未知 \*/

NETDEV\_SPEED\_TYPE\_STATIC = 1, /\* 静止 \*/

NETDEV\_SPEED\_TYPE\_SLOW = 2, /\* 慢速 \*/

NETDEV\_SPEED\_TYPE\_MEDIUM = 3, /\* 中速 \*/

NETDEV\_SPEED\_TYPE\_FAST = 4, /\* 快速 \*/

NETDEV\_SPEED\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_SPEED\_TYPE\_E;

### 结构化场景中非机动车相对画面运动方向枚举

typedef enum tagNETDEVImageDirection

{

NETDEV\_IMAGE\_DIRECTION\_UNKNOW = 0, /\* 未知 \*/

NETDEV\_IMAGE\_DIRECTION\_STATIC = 1, /\* 静止 \*/

NETDEV\_IMAGE\_DIRECTION\_UP = 2, /\* 向上 \*/

NETDEV\_IMAGE\_DIRECTION\_DOWN = 3, /\* 向下 \*/

NETDEV\_IMAGE\_DIRECTION\_LEFT = 4, /\* 向左 \*/

NETDEV\_IMAGE\_DIRECTION\_RIGHT = 5, /\* 向右 \*/

NETDEV\_IMAGE\_DIRECTION\_LEFTUP = 6, /\* 左上 \*/

NETDEV\_IMAGE\_DIRECTION\_LEFTDOWN = 7, /\* 左下 \*/

NETDEV\_IMAGE\_DIRECTION\_RIGHTUP = 8, /\* 右上 \*/

NETDEV\_IMAGE\_DIRECTION\_RIGHTDOWN = 9, /\* 右下 \*/

NETDEV\_IMAGE\_DIRECTION\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_IMAGE\_DIRECTION\_E;

### 通道权限枚举

typedef enum tagNETDEVChnPermission

{

NETDEV\_CHN\_PERMISSION\_LIVE = BIT0, /\* 实况和语音对讲 \*/

NETDEV\_CHN\_PERMISSION\_PTZ = BIT1, /\* 云台控制 \*/

NETDEV\_CHN\_PERMISSION\_PLAYBACK = BIT2, /\* 回放 \*/

NETDEV\_CHN\_PERMISSION\_MANU\_RECORD = BIT3, /\* 手动录像 \*/

NETDEV\_CHN\_PERMISSION\_LOCAL\_BACK = BIT4, /\* 本地备份 \*/

NETDEV\_CHN\_PERMISSION\_INVALID = 0XFFFFFFFF /\* 无效值 \*/

}NETDEV\_CHN\_PERMISSION\_TYPE\_E;

### 响应状态类型枚举

typedef enum tagNETDEVOrgResponseStatus

{

NETDEV\_ORG\_RESPONSE\_SUCCESS = 0, /\* 响应成功 \*/

NETDEV\_ORG\_RESPONSE\_FAIL = 1 /\* 响应失败 \*/

}NETDEV\_ORG\_RESPONSE\_STAUTE\_E;

### NAT类型枚举

typedef enum tagNETDEVNATType

{

NETDEV\_STUN\_TYPE\_UNKNOWN = 0, /\* 未识别 \*/

NETDEV\_STUN\_TYPE\_FAILURE = 1, /\* 失败 \*/

NETDEV\_STUN\_TYPE\_OPEN = 2 /\* 没有NAT，公共IP，没有防火墙 \*/

NETDEV\_STUN\_TYPE\_BLOCKED = 3, /\* UDP受阻 \*/

NETDEV\_STUN\_TYPE\_CONE\_NAT = 4, /\* 完全NAT \*/

NETDEV\_STUN\_TYPE\_RESTRICTED\_NAT = 5, /\* 受限NAT \*/

NETDEV\_STUN\_TYPE\_PORT\_RESTRICTED\_NAT = 6, /\* 端口受限NAT \*/

NETDEV\_STUN\_TYPE\_SYM\_NAT = 7, /\* 对称NAT \*/

NETDEV\_STUN\_TYPE\_SYM\_FIREWALL = 8 /\* 没有NAT，公共IP，但是对称的UDP防火墙 \*/

}NETDEV\_NAT\_TYPE\_E;

### 协议类型枚举

typedef enum tagNETDEVProtocolType

{

NETDEV\_PROTOCOL\_TYPE\_HTTP = 0,

NETDEV\_PROTOCOL\_TYPE\_HTTPS = 1,

NETDEV\_PROTOCOL\_TYPE\_RTSP = 2

}NETDEV\_PROTOCOL\_TYPE\_E;

### 组织查找模式枚举

typedef enum tagNETDEVOrgFindMode

{

NETDEV\_ORG\_FIND\_MODE\_ID = 0, /\* 组织ID \*/

NETDEV\_ORG\_FIND\_MODE\_TYPE = 1, /\* 组织类型 \*/

NETDEV\_ORG\_FIND\_MODE\_INVALID = 0XFF /\* 无效值 \*/

}NETDEV\_ORG\_FIND\_MODE\_E;

### 基本权限枚举

typedef enum tagNETDEVUserBasePermission

{

NETDEV\_USER\_BASE\_PER\_CONFIG = BIT0, /\* 配置\*/

NETDEV\_USER\_BASE\_PER\_UPGRADE = BIT1, /\* 升级 \*/

NETDEV\_USER\_BASE\_PER\_LOG = BIT2, /\* 日志查看和导出 \*/

NETDEV\_USER\_BASE\_PER\_REBOOT = BIT3 /\* 重启 \*/

}NETDEV\_USER\_BASE\_PERMISSION\_E;

### 硬盘模式

typedef enum tagNETDEVDiskModeType

{

NETDEV\_DISK\_MODE\_TYPE\_OLD = 0, /\* 旧存储格式硬盘\*/

NETDEV\_DISK\_MODE\_TYPE\_NEW = 1, /\* 新存储格式硬盘\*/

NETDEV\_DISK\_MODE\_TYPE\_INVALID = 0xFF /\* 无效值\*/

}NETDEV\_DISK\_MODE\_TYPE\_E;

### 能力集命令

typedef enum tagNETDEVCapabilityCommond

{

NETDEV\_CAP\_VIDEO\_ENCODE = 1, /\* 视频编码能力集参见#

NETDEV\_VIDEO\_STREAM\_CAP\_S。\*/

NETDEV\_CAP\_OSD = 2, /\* OSD参数能力集参见# NETDEV\_OSD\_CAP\_S。\*/

NETDEV\_CAP\_SMART = 3, /\* 智能能力集参见# NETDEV\_SMART\_CAP\_S。\*/

NETDEV\_CAP\_VIDEO\_ENCODE\_EX = 4, /\* 视频编码能力集（扩展）参见#

NETDEV\_VIDEO\_STREAM\_CAP\_EX\_S。\*/

NETDEV\_CAP\_IMAGE = 5, /\* 图像参数能力集参见#NETDEV\_IMAGE\_CAP\_S。\*/

NETDEV\_CAP\_AUDIO = 6, /\* 音频能力集参见NETDEV\_AUDIO\_CAP\_S \*/

NETDEV\_CAP\_VIDEO\_SNAPSHOT = 7, /\* 视频输入通道的抓图参数能力参见#NETDEV\_VIDEO\_SNAP\_CAP\_S 目前仅IPC支持\*/

NETDEV\_CAP\_FACE\_RECOGNAZE = 10, /\* 人脸识别能力目前仅NVR支持, 参见NETDEV\_DEV\_FACE\_CAP\_INFO\_S \*/

NETDEV\_CAP\_DEV\_NIC = 11, /\* 设备网卡能力集, 参见

NETDEV\_NIC\_CAP\_INFO\_S \*/

NETDEV\_CAP\_DEV\_EXCEPTION\_ALARM = 12, /\* 设备异常告警能力集, 参见#

NETDEV\_EXCP\_ALARM\_CAP\_INFO\_S \*/

NETDEV\_CAP\_CHANNELS\_ALARM = 13, /\* 通道告警能力集, 参见

NETDEV\_CHN\_ALARM\_CAP\_INFO\_S （单通道IPC对应SDK通道号传入参数；多通道IPC对应SDK通道号传入参数+IPC实际通道号；NVR下对应通道号传入实际通道号）\*/

NETDEV\_CAP\_SYS = 14, /\* 系统能力集参见

NETDEV\_SYS\_CAPABILITY\_S \*/

NETDEV\_CAP\_INVALID = 0Xff

}NETDEV\_CAPABILITY\_COMMOND\_E;

### OSD日期格式能力集

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;

### 时间同步方式枚举

typedef enum tagNETDEVTimeSynmode

{

NETDEV\_TIME\_SYNCMODE\_ACCEPT\_ALL\_SERVER = 0, /\* 接受所有服务器同步时间 \*/

NETDEV\_TIME\_SYNCMODE\_SYNC\_COMPUTER\_SYSTEM = 1, /\* 同步计算机系统时间 \*/

NETDEV\_TIME\_SYNCMODE\_SYNC\_PHOTO\_SERVER = 2, /\* 同步照片服务器时间 \*/

NETDEV\_TIME\_SYNCMODE\_SYNC\_NTP\_SERVER = 3, /\* 同步NTP服务器时间 \*/

NETDEV\_TIME\_SYNCMODE\_SYNC\_MANAGE\_SERVER = 4, /\* 同步管理服务器时间（非Onvif接入） \*/

NETDEV\_TIME\_SYNCMODE\_SYNC\_MANAGE\_SERVER\_ONVIF= 5,/\*同步管理服务器时间（Onvif接入） \*/

NETDEV\_TIME\_SYNCMODE\_SYNC\_CLOUD\_SERVER = 6, /\* 同步云服务器时间 \*/

NETDEV\_TIME\_SYNCMODE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_TIME\_SYNCMODE\_E;

### 码流数据回调函数

typedef enum tagNETDEStreamDataCBType

{

NETDEV\_STREAM\_CB\_TYPE\_PARSE = 1, /\* 拼帧后音/视频回调 \*/

NETDEV\_STREAM\_CB\_TYPE\_DECODE = 2, /\* 解码后音/视频回调 \*/

NETDEV\_STREAM\_CB\_TYPE\_DECODE\_V30 = 3, /\* 解码后音/视频回调,支持GPU解码 \*/

NETDEV\_STREAM\_CB\_TYPE\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_STREAM\_DATA\_CB\_TYPE\_E;

### 人数统计类型

typedef enum tagNETDEVStatisticType

{

NETDEV\_STATISTIC\_TYPE\_ALL = 0, /\* 总人数统计 \*/

NETDEV\_STATISTIC\_TYPE\_ENTER = 1, /\* 进入人数统计 \*/

NETDEV\_STATISTIC\_TYPE\_LEAVE = 2, /\* 离开人数统计 \*/

NETDEV\_STATISTIC\_TYPE\_INVALID = 0xFFFF /\* 无效值 \*/

}NETDEV\_STATISTIC\_TYPE\_E;

### 监听类型枚举

typedef enum tagNETDEVListenType

{

NETDEV\_FACE\_SNAPSHOT\_TYPE = 1, /\* 人脸抓拍 \*/

NETDEV\_DISCOVERY\_TYPE = 2, /\* 设备搜索 \*/

NETDEV\_LISTEN\_TYPE\_INVALID = 0xffff

}NETDEV\_LISTEN\_TYPE\_E;

### 告警支持类型枚举

typedef enum tagNETDEVAlarmTypeV30

{

NETDEV\_ALARM\_RYPE\_DEV\_STATUS = BIT0, /\* 设备状态类型告警 \*/

NETDEV\_ALARM\_RYPE\_COMM\_ALARM = BIT1, /\* 监控业务类告警 \*/

NETDEV\_ALARM\_RYPE\_INTEL\_ALARM = BIT2, /\* 泛智能告警 \*/

NETDEV\_ALARM\_RYPE\_SMART\_ALARM = BIT3, /\* 智能类告警 \*/

NETDEV\_ALARM\_RYPE\_FACE\_RECOGNITION = BIT4, /\* 人脸识别 NETDEV\_SetPersonAlarmCallBack \*/

NETDEV\_ALARM\_RYPE\_STRUCTURED\_DATA = BIT5, /\* 结构化数 NETDEV\_SetStructAlarmCallBack \*/

NETDEV\_ALARM\_RYPE\_VEHICLE\_RECOGNITION = BIT6, /\* 车牌识别 NETDEV\_SetVehicleAlarmCallBack \*/

NETDEV\_ALARM\_RYPE\_TRAFFIC\_DATA = BIT7, /\* 交通数据 (暂未支持) \*/

NETDEV\_ALARM\_RYPE\_HYPERSENSITIVE\_DATA = BIT8, /\* 超感数据 (暂未支持) \*/

NETDEV\_ALARM\_RYPE\_RESOURCE\_CHANGE = BIT9, /\* 资源变更 NETDEV\_SetResChangeEventCallBack \*/

NETDEV\_ALARM\_RYPE\_PERSON\_VERIFICATION = BIT10, /\* 人员核验 NETDEV\_SetAlarmFGCallBack \*/

NETDEV\_ALARM\_RYPE\_PARKING\_IDENTIFICATION = BIT11, /\* 车场抓拍 NETDEV\_SetParkEventCallBack \*/

NETDEV\_ALARM\_RYPE\_FIREALARM\_DATA = BIT12, /\* 火点告警 \*/

NETDEV\_ALARM\_RYPE\_ALARM\_PICTURE\_DATA = BIT13, /\* 告警图片数据 NETDEV\_SetPicAlarmCallBack \*/

NETDEV\_ALARM\_RYPE\_PEOPLE\_COUNT = BIT14, /\* 人数统计 NETDEV\_SetPeopleCountAlarmCallBack \*/

NETDEV\_ALARM\_RYPE\_HEATMAP\_DATA = BIT16, /\* 热度图数据 (暂未支持) \*/

NETDEV\_ALARM\_RYPE\_PLAYBOX\_STATUS = BIT17, /\* 播放盒状态 (暂未支持) \*/

NETDEV\_ALARM\_RYPE\_PLAYBOX\_MANAGEMENT = BIT18, /\* 播放盒管理设备在线状态(暂未支持).\*/

NETDEV\_ALARM\_RYPE\_INVALID = 0xFF

}NETDEV\_ALARM\_TYPE\_V30\_E;

### 号牌颜色类型枚举

typedef enum tagNETDEVTMSCarPlateColor

{

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_WHITE = 0, /\* 白色 \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_YELLOW = 1, /\* 黄色 \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_BLUE = 2, /\* 蓝色 \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_BLACK = 3, /\* 黑色 \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_OTHER = 4, /\* 其它颜色 \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_GREEN = 5, /\* 绿色，农用车 \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_RED = 6, /\* 红色 \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_YELLOW\_AND\_GREEN = 7, /\* 黄绿双色 \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_GRADIENT\_GREEN = 8， /\* 渐变绿色 \*/

NETDEV\_TMS\_CAR\_PLATE\_COLOR\_INVALID = 0xFFFF /\* 无效值 \*/

}NETDEV\_TMS\_CAR\_PLATE\_COLOR\_E;

### 按位查询录像类型枚举

typedef enum tagNETDEVRecordSearchType

{

NETDEV\_RECORD\_SEARCH\_TYPE\_ALL = 0x00000000, /\* 查找时使用，全部录像类型 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_MANUL = 0x00000001, /\* 手动录像 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_EVENT = 0x00000002, /\* 通用类告警（NVR暂不支持） \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_MOTION = 0x00000004, /\* 运动检测 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_ALARMIN = 0x00000008, /\* 输入开关量 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_VIDEO\_LOSS= 0x00000010, /\* 视频丢失 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_AUDIO\_DETECT= 0x00000020, /\* 音频检测 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_COMMON = 0x00000080, /\* 计划录像(常规录像) \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_FACE\_DETECT = 0x00000100, /\* 人脸检测 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_LINE\_DETECT = 0x00000200, /\* 越界检测 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_FIELD\_DETECT = 0x00000400, /\* 区域入侵 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_FOCUS\_DETECT = 0x00000800, /\* 虚焦检测 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_SCENE\_CHANGE= 0x00001000, /\* 场景变更 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_SMART\_TRACK = 0x00002000, /\* 智能运动跟踪事件 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_URGENT\_BELL = 0x00004000, /\* 紧急铃 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_REMAIN\_ARTICLE = 0x00020000, /\* 物品遗留 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_MOVE\_ARTICLE = 0x00040000, /\* 物品搬移 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_HUMAN\_DETECT = 0x00200000, /\* 人形检测 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_ELEVATOR\_ENTRANCE\_DETECT= 0x20000000, /\* 入梯检测 \*/

NETDEV\_RECORD\_SEARCH\_TYPE\_SMART\_RECORD = 0x20263F20, /\* 所有智能类告警 0010 0000 0010 0110 0011 1111 0010 0000\*/

NETDEV\_RECORD\_SEARCH\_TYPE\_INVALID = 0XFFFFFFFF /\* 无效值 \*/

}NETDEV\_RECORD\_SEARCH\_TYPE\_E;

### 肤色枚举

typedef enum tagNETDEVSkinColorType

{

NETDEV\_SKINCOLOR\_TYPE\_UNKNOW = 0, /\* 未知 \*/

NETDEV\_SKINCOLOR\_TYPE\_WHITE = 2011, /\* 白皮肤 \*/

NETDEV\_SKINCOLOR\_TYPE\_BLACK = 2012, /\* 黑皮肤 \*/

NETDEV\_SKINCOLOR\_TYPE\_YELLOW = 2013, /\* 黄皮肤 \*/

NETDEV\_SKINCOLOR\_TYPE\_BROWN = 2014, /\* 棕皮肤 \*/

NETDEV\_SKINCOLOR\_TYPE\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_SKINCOLOR\_TYPE\_E;

### 情绪情况

typedef enum tagNETDEVEmotionFlag

{

NETDEV\_EMOTION\_FLAG\_UNKNOW = 0, /\* 未知 \*/

NETDEV\_EMOTION\_FLAG\_ANGER = 1, /\* 生气的 \*/

NETDEV\_EMOTION\_FLAG\_CALM = 2, /\* 平静的 \*/

NETDEV\_EMOTION\_FLAG\_CONFUSED = 3, /\* 迷茫的 \*/

NETDEV\_EMOTION\_FLAG\_ABHORRENT = 4, /\* 厌恶的 \*/

NETDEV\_EMOTION\_FLAG\_HAPPY = 5, /\* 高兴的 \*/

NETDEV\_EMOTION\_FLAG\_SAD = 6, /\* 悲伤的 \*/

NETDEV\_EMOTION\_FLAG\_AFRAID = 7, /\* 害怕的 \*/

NETDEV\_EMOTION\_FLAG\_AMAZED = 8, /\* 吃惊的 \*/

NETDEV\_EMOTION\_FLAG\_SQUINT = 9, /\* 眯眼的 \*/

NETDEV\_EMOTION\_FLAG\_SCREAM = 10, /\* 尖叫的 \*/

NETDEV\_EMOTION\_FLAG\_OTHER = 11, /\* 其他 \*/

NETDEV\_EMOTION\_FLAG\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_EMOTION\_FLAG\_E;

### 微笑标志

typedef enum tagNETDEVSmileFlag

{

NETDEV\_SMILE\_FLAG\_UNKNOW = 0, /\* 未知 \*/

NETDEV\_SMILE\_FLAG\_NO = 1, /\* 不微笑 \*/

NETDEV\_SMILE\_FLAG\_YES = 2, /\* 微笑 \*/

NETDEV\_SMILE\_FLAG\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_SMILE\_FLAG\_E;

### 胡子标志

typedef enum tagNETDEVBeardFlag

{

NETDEV\_BEARD\_FLAG\_UNKNOW = 0, /\* 未知 \*/

NETDEV\_BEARD\_FLAG\_UNEXIST = 1, /\* 没胡子 \*/

NETDEV\_BEARD\_FLAG\_EXIST = 2, /\* 有胡子 \*/

NETDEV\_BEARD\_FLAG\_INVALID = 0xFF /\* 无效值 \*/

}NETDEV\_BEARD\_FLAG\_E;

### 目标类型枚举

typedef enum tagNETDEVObjectType

{

NETDEV\_OBJECT\_TYPE\_FACE = 1, /\* 人脸 \*/

NETDEV\_OBJECT\_TYPE\_PERSON = 2, /\* 人员 \*/

NETDEV\_OBJECT\_TYPE\_NON\_MOTOR = 3, /\* 非机动车 \*/

NETDEV\_OBJECT\_TYPE\_MOTOR = 4, /\* 机动车 \*/

NETDEV\_OBJECT\_TYPE\_UNKNOWN = 255, /\* 未知/未识别的目标类型 \*/

NETDEV\_OBJECT\_TYPE\_INVALID = 0xffff /\* 无效值 \*/

}NETDEV\_OBJECT\_TYPE\_E;

### 规则类型枚举

typedef enum tagNETDEVRuleType

{

NETDEV\_RULE\_TYPE\_INTRUSION\_DETECTION = 0, /\* 区域入侵 \*/

NETDEV\_RULE\_TYPE\_CROSSLINE\_DETECTION = 1, /\* 越界检测 \*/

NETDEV\_RULE\_TYPE\_LEAVE\_ZONE = 2, /\* 离开区域 \*/

NETDEV\_RULE\_TYPE\_ENTER\_ZONE = 3, /\* 进入区域 \*/

NETDEV\_RULE\_TYPE\_ELEVATOR\_ENTRANCE\_DETECTION = 4, /\* 入梯检测报警 \*/

NETDEV\_RULE\_TYPE\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_RULE\_TYPE\_E;

### 规则触发类型枚举

typedef enum tagNETDEVRuleTriggerType

{

NETDEV\_RULE\_TRIGGER\_TYPE\_NO\_TRIGGER = 0, /\* 无触发类型 \*/

NETDEV\_RULE\_TRIGGER\_TYPE\_DUPLEX = 1, /\* 双向触发 \*/

NETDEV\_RULE\_TRIGGER\_TYPE\_CLOCKWISE = 2, /\* 顺时针触发 \*/

NETDEV\_RULE\_TRIGGER\_TYPE\_ANTIC\_CLOCKWISE = 3, /\* 逆时针触发 \*/

NETDEV\_RULE\_TRIGGER\_TYPE\_ACCESS = 4, /\* 进入触发 \*/

NETDEV\_RULE\_TRIGGER\_TYPE\_LEAVE = 5, /\* 离开触发 \*/

NETDEV\_RULE\_TRIGGER\_TYPE\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_RULE\_TRIGGER\_TYPE\_E;

### 目标记录类型

typedef enum tagNETDEVObjectRecordType

{

NETDEV\_OBJECT\_RECORD\_TYPE\_CROSSLINE\_DETECTION = 0, /\* 越界检测 \*/

NETDEV\_OBJECT\_RECORD\_TYPE\_INTRUSION\_DETECTION = 1, /\* 区域入侵 \*/

NETDEV\_OBJECT\_RECORD\_TYPE\_ENTER\_ZONE = 2, /\* 进入区域 \*/

NETDEV\_OBJECT\_RECORD\_TYPE\_LEAVE\_ZONE = 3, /\* 离开区域 \*/

NETDEV\_OBJECT\_RECORD\_TYPE\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_OBJECT\_RECORD\_TYPE\_E;

### 人数统计告警类型

typedef enum tagNETDEVPeopleCountAlarmType

{

NETDEV\_PEOPLE\_COUNT\_ALARM\_TYPE\_AREA\_RULE = 0, /\* 区域规则 \*/

NETDEV\_PEOPLE\_COUNT\_ALARM\_TYPE\_LINE\_RULE = 1, /\* 绊线规则 \*/

NETDEV\_PEOPLE\_COUNT\_ALARM\_TYPE\_CROWD\_DENSITY = 2, /\* 人员密度 \*/

NETDEV\_PEOPLE\_COUNT\_ALARM\_TYPE\_INVALID = 0xFFFF /\* 无效值 \*/

}NETDEV\_PEOPLE\_COUNT\_ALARM\_TYPE\_E;

### 人员密度报警统计类型

typedef enum tagNETDEVCrowdDensityStatisticType

{

NETDEV\_CROWD\_DENSITY\_STATISTIC\_TYPE\_DENSITY = 1, /\* 人员密度统计 \*/

NETDEV\_CROWD\_DENSITY\_STATISTIC\_TYPE\_STRANDED = 2, /\* 滞留人数统计 \*/

NETDEV\_CROWD\_DENSITY\_STATISTIC\_TYPE\_INVALID = 0xff /\* 无效值 \*/

}NETDEV\_CROWD\_DENSITY\_STATISTIC\_TYPE\_E;

## 错误码列表

### 通用错误码

|  |  |  |
| --- | --- | --- |
| **错误码** | **错误码值** | **含义** |
| NETDEV\_E\_FAILED | -1 | 失败 |
| NETDEV\_E\_SUCCEED | 0 | 成功 |
| NETDEV\_E\_SVC\_FAILED | 1 | 服务器失败错误码 |
| NETDEV\_E\_NO\_USER | 5 | 用户不存在 |
| NETDEV\_E\_NO\_RESULT | 11 | 查无结果 |
| NETDEV\_E\_NOENOUGH\_BUF | 12 | 缓冲区太小 |
| NETDEV\_E\_SDK\_SOCKET\_LSN\_FAIL | 13 | 创建socket |
| NETDEV\_E\_INIT\_MUTEX\_FAIL | 14 | 初始化锁失败 |
| NETDEV\_E\_INIT\_SEMA\_FAIL | 15 | 初始化信号量失败 |
| NETDEV\_E\_ALLOC\_RESOURCE\_ERROR | 16 | SDK资源分配错误 |
| NETDEV\_E\_HAVEDATA | 17 | 还有数据 |
| NETDEV\_E\_NEEDMOREDATA | 18 | 需要更多数据 |
| NETDEV\_E\_TRANSFILE\_FAIL | 19 | 文件传输失败 |
| NETDEV\_E\_DEVICE\_TYPE\_ERR | 20 | 不支持的设备类型 |
| NETDEV\_E\_NONCE\_TIMEOUT | 21 | nonce过期 |
| NETDEV\_E\_INNER\_ERR | 22 | 系统内部错误 |
| NETDEV\_E\_PUBLICKEYFAIL | 23 | 公钥匹配失败 |
| NETDEV\_E\_BINDNOTIFY\_FAIL | 24 | 绑定告警失败 |
| NETDEV\_E\_SYSCALL\_FALIED | 100 | 系统函数调用失败请查看errno |
| NETDEV\_E\_NULL\_POINT | 101 | 空指针 |
| NETDEV\_E\_INVALID\_PARAM | 102 | 无效参数 |
| NETDEV\_E\_INVALID\_MODULEID | 103 | 无效模块ID |
| NETDEV\_E\_INVALID\_HANDLE | 104 | 无效的句柄 |
| NETDEV\_E\_NO\_MEMORY | 105 | 内存分配失败 |
| NETDEV\_E\_FILE\_NO\_EXIST | 106 | 文件等不存在 |
| NETDEV\_E\_NO\_DEV | 107 | 设备不存在 |
| NETDEV\_E\_NO\_FIT\_LOG | 108 | 符合条件的日志不存在 |
| NETDEV\_E\_BUSY | 109 | busy状态 |
| NETDEV\_E\_TIMER\_REG\_FAILED | 110 | 注册定时器失败 |
| NETDEV\_E\_COMMON\_FAILED | 111 | 通用错误 |
| NETDEV\_E\_CMD\_NOT\_SUPPORT | 112 | 命令不支持 |
| NETDEV\_E\_NOT\_SUPPORT | 113 | 设备不支持该功能 |
| NETDEV\_E\_TIMEOUT | 114 | 超时 |
| NETDEV\_E\_MSG\_ERR | 115 | 消息不匹配 |
| NETDEV\_E\_MODULE\_INEXIST | 116 | 模块不存在 |
| NETDEV\_E\_SOCKET\_RECV\_ERR | 117 | 消息接收失败 |
| NETDEV\_E\_DECODE\_IE\_FAILED | 118 | 获取消息IE失败 |
| NETDEV\_E\_ENCODE\_IE\_FAILED | 119 | 添加消息IE失败 |
| NETDEV\_E\_SDK\_NOINTE\_ERROR | 120 | SDK未初始化 |
| NETDEV\_E\_ALREDY\_INIT\_ERROR | 121 | SDK已经初始化 |
| NETDEV\_E\_DEVICE\_FACTURER\_ERR | 122 | 不支持的设备厂商 |
| NETDEV\_E\_NAME\_EXIST | 123 | 名称已存在 |
| NETDEV\_E\_GET\_CFG\_FAILED | 124 | 获取配置信息出错 |
| NETDEV\_E\_SET\_CFG\_FAILED | 125 | 设置配置信息出错 |
| NETDEV\_E\_CHANNEL\_OVER\_SPEC | 126 | 通道数超规格 |
| NETDEV\_E\_CALL\_DRV\_COMMON | 127 | 调用驱动通用失败 |
| NETDEV\_E\_TOTAL\_QUOTA\_FULL | 128 | 可分配的配额空间不足 |
| NETDEV\_E\_CALL\_DB\_COMMON | 129 | 调用数据库通用失败 |
| NETDEV\_E\_NEED\_MORE\_MEMORY | 130 | 内存分配不足 |
| NETDEV\_E\_T2U\_CONNECT\_FAILED | 131 | T2U连接失败 |
| NETDEV\_E\_FUNC\_IS\_INITIALIZING | 132 | 功能正在初始化中 |
| NETDEV\_E\_CONNECT\_ERROR | 200 | 创建连接失败 |
| NETDEV\_E\_SEND\_MSG\_ERROR | 201 | 发送消息失败 |
| NETDEV\_E\_DECODE\_RSP\_ERROR | 202 | 解析响应消息失败 |
| NETDEV\_E\_NONSUPPORT | 203 | 该功能函数未实现 |
| NETDEV\_E\_JSON\_ERROR | 204 | Json通用错误 |
| NETDEV\_E\_NORESULT | 205 | 查询结果为空 |
| NETDEV\_E\_SOCKET\_RECV\_ERROR | 206 | Socket接收消息失败 |
| NETDEV\_E\_CREATE\_THREAD\_FAIL | 207 | 创建线程失败 |
| NETDEV\_E\_RESCODE\_NO\_EXIST | 208 | 资源编码不存在 |
| NETDEV\_E\_MSG\_DATA\_INVALID | 209 | 消息内容错误 |
| NETDEV\_E\_JSON\_NO\_IMAGE | 210 | 图片数据为空 |
| NETDEV\_E\_IMAGE\_SIZE\_BEYOND\_THE\_LIMIT | 211 | 图片大小超出限制 |
| NETDEV\_E\_MANAGETYPE\_NONSUPPORT | 403 | 不支持改客户端/平台类型 |
| NETDEV\_E\_PUBLICKEYFAIL | 60068 | 公钥匹配失败 |

### 流媒体错误码

|  |  |  |
| --- | --- | --- |
| **错误码** | **错误码值** | **含义** |
| NETDEV\_E\_PLAYER\_FAIL | 1001 | 执行失败 |
| NETDEV\_E\_PLAYER\_INVALID\_PARAM | 1002 | 输入参数非法 |
| NETDEV\_E\_PLAYER\_NO\_MEMORY | 1003 | 系统内存不足 |
| NETDEV\_E\_PLAYER\_SOCKET\_FAIL | 1004 | 创建SOCKET失败 |
| NETDEV\_E\_PLAYER\_RECV\_FAIL | 1005 | 接收失败 |
| NETDEV\_E\_PLAYER\_RECV\_ZERO | 1006 | 接收为零 |
| NETDEV\_E\_PLAYER\_NOT\_SUPPORT | 1007 | 功能暂不支持 |
| NETDEV\_E\_PLAYER\_CREATETHREAD\_FAILED | 1008 | 创建线程失败 |
| NETDEV\_E\_PLAYER\_OPENDL\_FAILED | 1009 | 加载动态库失败 |
| NETDEV\_E\_PLAYER\_SYMDL\_FAILED | 1010 | 获取动态方法失败 |
| NETDEV\_E\_PLAYER\_SEND\_FAILED | 1011 | 发送失败 |
| NETDEV\_E\_PLAYER\_EACCES | 1012 | 创建文件权限不足 |
| NETDEV\_E\_PLAYER\_FILE\_NOT\_FIND | 1013 | 读文件未找到 |
| NETDEV\_E\_PLAYER\_LOG\_CLOSE | 1014 | 日志关闭 |
| NETDEV\_E\_PLAYER\_MEDIA\_EXCEPTION | 1017 | 内部处理异常 |
| NETDEV\_E\_PLAYER\_SYS\_FAIL | 1018 | 系统通用错误 |
| NETDEV\_E\_PLAYER\_INIT\_DONE | 1019 | 已经初始化 |
| NETDEV\_E\_PLAYER\_SYS\_RES\_FAILED | 1020 | 创建系统资源失败 |
| NETDEV\_E\_PLAYER\_INVALID\_IP | 1021 | IP错误 |
| NETDEV\_E\_PLAYER\_EZSTREAMER\_FULL | 1022 | Stream规格满 |
| NETDEV\_E\_PLAYER\_VOD\_OVER\_ABILITY | 1023 | 一体机流规格满 |
| NETDEV\_E\_PLAYER\_STREAM\_IN\_PROCESS | 1024 | 流存储已经处理中 |
| NETDEV\_E\_PLAYER\_NO\_SPARE\_SESSION | 1025 | 无空闲会话 |
| NETDEV\_E\_PLAYER\_NEED\_AUTHENTICATE | 1026 | 需要鉴权 |
| NETDEV\_E\_PLAYER\_GET\_AUTHENTICATE\_FAID | 1027 | 获取鉴权失败 |
| NETDEV\_E\_PLAYER\_MAKE\_AUTHENTICATE\_FAID | 1028 | 制作鉴权失败 |
| NETDEV\_E\_PLAYER\_AUTHENTICATEINFO\_DIFF | 1029 | 鉴权信息不一致 |
| NETDEV\_E\_PLAYER\_SESSION\_CLOSED | 1030 | 会话已关闭 |
| NETDEV\_E\_FAIL\_TO\_INIT\_EZPLAYER | 1257 | 初始化播放器失败 |
| NETDEV\_E\_FAIL\_TO\_ALLOC\_PORT\_RES | 1258 | 分配播放通道资源失败 |
| NETDEV\_E\_FAIL\_TO\_GET\_PORT\_RES | 1259 | 获得播放通道资源失败 |
| NETDEV\_E\_BUFFER\_QUEUE\_FULL | 1260 | 缓存队列已满 |
| NETDEV\_E\_BUFFER\_QUEUE\_EMPTY | 1261 | 缓存队列空 |
| NETDEV\_E\_OPEN\_FILE\_FAILED | 1262 | 打开文件失败 |
| NETDEV\_E\_FILE\_READ\_END | 1263 | 文件已经读取完毕 |
| NETDEV\_E\_FILE\_DISKSPACE\_FULL | 1264 | 磁盘空间满 |
| NETDEV\_E\_FILE\_READ\_FAIL | 1265 | 读取失败 |
| NETDEV\_E\_MCM\_MIC\_NOT\_EXIST | 1266 | 麦克风不存在 |
| NETDEV\_E\_TS\_PACKET\_IN\_THE\_ROUGH | 1267 | TS打包未完成 |
| NETDEV\_E\_FILE\_RECORD\_FINISH | 1268 | 录像保存完毕 |
| NETDEV\_E\_VIDEO\_RESOLUTION\_CHANGE | 1269 | 分辨率发生变化 |
| NETDEV\_E\_VIDEO\_RECORD\_PART | 1270 | 录像分段 |
| NETDEV\_E\_FAIL\_TO\_OPEN\_STREAM | 1513 | 启动媒体流播放失败 |
| NETDEV\_E\_FAIL\_TO\_CLOSE\_STREAM | 1514 | 关闭媒体流播放失败 |
| NETDEV\_E\_FAIL\_TO\_RECV\_DATA | 1515 | 网络故障导致接收数据失败 |
| NETDEV\_E\_FAIL\_TO\_PROCESS\_MEDIA\_DATA | 1516 | 媒体数据处理失败 |
| NETDEV\_E\_NOT\_START\_PLAY | 1517 | 播放通道未开始播放操作 |
| NETDEV\_E\_FAIL\_TO\_INPUT\_DATA | 1518 | 输入媒体流数据失败 |
| NETDEV\_E\_INPUTDATA\_BUFFER\_FULL | 1519 | 输入数据缓存满 |
| NETDEV\_E\_FAIL\_TO\_SET\_PROCESS\_DATA\_CB | 1520 | 设置媒体流数据回调函数失败 |
| NETDEV\_E\_VOICE\_RUNNING | 1521 | 语音业务运行过程中出错 |
| NETDEV\_E\_FAIL\_TO\_OPEN\_VOICE\_SVC | 1522 | 启动语音业务失败 |
| NETDEV\_E\_FAIL\_TO\_CLOSE\_VOICE\_SVC | 1523 | 关闭语音业务失败 |
| NETDEV\_E\_UNKNOWN\_STREAM\_TYPE | 1524 | 未知媒体流 |
| NETDEV\_E\_PACKET\_LOSE | 1525 | 丢包 |
| NETDEV\_E\_NEED\_MORE\_PACKET | 1526 | 拼包未完成需要更多包 |
| NETDEV\_E\_FAIL\_TO\_CREATE\_DECODE | 1527 | 创建解码器失败 |
| NETDEV\_E\_FAIL\_TO\_DECODE | 1528 | 解码失败 |
| NETDEV\_E\_RECV\_DATA\_NOTENOUGH | 1529 | 接收数据不足 |
| NETDEV\_E\_RENDER\_RES\_FULL | 1530 | 显示资源满 |
| NETDEV\_E\_RENDER\_RES\_NOT\_EXIST | 1531 | 显示资源不存在 |
| NETDEV\_E\_CREATE\_DEV\_FAILED | 1532 | 资源创建失败 |
| NETDEV\_E\_AUDIO\_RES\_NOT\_EXIST | 1533 | 音频资源不存在 |
| NETDEV\_E\_IHW265D\_NEED\_MORE\_BITS | 1534 | 解码器需要更多数据 |
| NETDEV\_E\_FAIL\_TO\_CREATE\_ENCODE | 1535 | 创建编码器失败 |
| NETDEV\_E\_CAPTURE\_RES\_EXIST | 1536 | 采集资源不存在 |
| NETDEV\_E\_RECORD\_STARTED | 1537 | 录像已打开 |
| NETDEV\_E\_NEED\_WAIT\_DECODEC | 1538 | 未解码完成需要等待 |
| NETDEV\_E\_MORE\_DATA\_NEED\_PACKET | 1539 | 数据过多还需要继续打包 |
| NETDEV\_E\_AAC\_LC\_DECODE\_FAIL | 1540 | AAC\_LC解码失败 |
| NETDEV\_E\_RENDER\_SURFACELOST | 1541 | 显示表面丢失 |
| NETDEV\_E\_PLAYBACK\_FINISH | 1542 | 回放结束 |
| NETDEV\_E\_FILE\_ENCRYPED | 1543 | 文件已加密 |
| NETDEV\_E\_SCRAMBLING\_INFO\_FAILED | 1544 | 加扰信息异常 |
| NETDEV\_E\_LIVE\_EXISTED | 2000 | 实况业务已经建立 |
| NETDEV\_E\_LIVE\_INPUT\_NOT\_READY | 2001 | 媒体流未准备就绪 |
| NETDEV\_E\_LIVE\_OUTPUT\_BUSY | 2002 | 实况业务显示资源忙 |
| NETDEV\_E\_LIVE\_CB\_NOTEXIST | 2003 | 实况控制块不存在 |
| NETDEV\_E\_LIVE\_STREAM\_FULL | 2004 | 实况流资源已满 |
| NETDEV\_E\_LIVE\_NET\_FAILED | 2005 | 会话网络错误 |
| NETDEV\_E\_LIVE\_NET\_TIMEOUT | 2006 | 会话网络超时 |
| NETDEV\_E\_LIVE\_SHAKE\_FAILED | 2007 | 会话交互错误 |
| NETDEV\_E\_LIVE\_AUTH\_FAILED | 2008 | 鉴权失败 |
| NETDEV\_E\_LIVE\_INNER\_ERROR | 2009 | 设备侧内部处理错误 |
| NETDEV\_E\_LIVE\_INNER\_TIMEOUT | 2010 | 内部处理超时 |
| NETDEV\_E\_LIVE\_KEEP\_ALIVE\_FAILED | 2011 | 保活失败 |
| NETDEV\_E\_LIVE\_SESSION\_NOT\_EXIST | 2012 | 会话不存在 |
| NETDEV\_E\_LIVE\_NOT\_ENOUGH\_BANDWIDTH2 | 2013 | 带宽不足 |
| NETDEV\_E\_LIVE\_REALPLAY\_ESTABLISHED | 2014 | 实况业务已经建立 |
| NETDEV\_E\_LIVE\_REALPLAY\_RES\_BUSY | 2015 | 实况业务显示资源忙 |
| NETDEV\_E\_LIVE\_MULTICAST\_DISABLED | 2016 | 组播使能关闭 |
| NETDEV\_E\_LIVE\_MULTICAST\_PORT\_OCCUPIED | 2017 | 组播端口已被占用 |
| NETDEV\_E\_LIVE\_MULTICAST\_PORT\_EXHAUSTED | 2018 | 组播端口已耗尽 |
| NETDEV\_E\_LIVE\_MULTICAST\_USER\_NOT\_EXIST | 2019 | 组播用户不存在 |
| NETDEV\_E\_LIVE\_CHANNEL\_NOT\_ONLINE | 2020 | 通道不在线 |
| NETDEV\_E\_LIVE\_TALKBACK\_ENCODED\_INVALID | 2021 | 语音对讲资源编码无效 |
| NETDEV\_E\_LIVE\_VOICE\_RES\_USED\_BY\_TALKBACK | 2022 | 语音资源已被对讲使用 |
| NETDEV\_E\_LIVE\_TALKBACK\_EXISTS | 2023 | 语音对讲已存在 |
| NETDEV\_E\_LIVE\_VOICE\_WORK\_NOT\_EXIST | 2024 | 语音业务不存在 |
| NETDEV\_E\_LIVE\_TALKBACK\_TIMEOUT | 2025 | 建立语音对讲业务超时 |
| NETDEV\_E\_LIVE\_TALKBACK\_ERROR | 2026 | 语音对讲失败 |
| NETDEV\_E\_LIVE\_UNDEFINED\_ERROR | 2027 | 未定义错误 |
| NETDEV\_E\_LIVE\_BAD\_REQUEST | 2028 | 错误的请求 |
| NETDEV\_E\_LIVE\_UNAUTHORIZED | 2029 | 未通过认证 |
| NETDEV\_E\_LIVE\_PAYMENT\_REQUIRED | 2030 | 需要付费 |
| NETDEV\_E\_LIVE\_FORIBIDDEN | 2031 | 禁止 |
| NETDEV\_E\_LIVE\_METHOD\_NOT\_ALLOWED | 2032 | 不允许该方法 |
| NETDEV\_E\_LIVE\_NOT\_ACCEPTABLE | 2033 | 不接受 |
| NETDEV\_E\_LIVE\_PROXY\_REQUIRED | 2034 | 代理需要认证 |
| NETDEV\_E\_LIVE\_REQUEST\_TIMEOUT | 2035 | 请求超时 |
| NETDEV\_E\_LIVE\_GONE | 2036 | 不在服务器 |
| NETDEV\_E\_LIVE\_LENGTH\_REQUIRED | 2037 | 需要长度 |
| NETDEV\_E\_LIVE\_PRECONDITION\_FAILED | 2038 | 预处理失败 |
| NETDEV\_E\_LIVE\_ENTITY\_TOO\_LARGE | 2039 | 请求实体过长 |
| NETDEV\_E\_LIVE\_URI\_TOO\_LARGE | 2040 | 请求-URI过长 |
| NETDEV\_E\_LIVE\_UNSUPPORTED\_TYPE | 2041 | 媒体类型不支持 |
| NETDEV\_E\_LIVE\_NOT\_UNDERSTOOD | 2042 | 不理解此参数 |
| NETDEV\_E\_LIVE\_CONFERENCE\_NOT\_FOUND | 2043 | 找不到会议 |
| NETDEV\_E\_LIVE\_NOT\_ENOUGH\_BANDWIDTH | 2044 | 带宽不足 |
| NETDEV\_E\_LIVE\_SESSION\_NOT\_FOUND | 2045 | 找不到会话 |
| NETDEV\_E\_LIVE\_METHOD\_NOT\_VALID | 2046 | 此状态下此方法无效 |
| NETDEV\_E\_LIVE\_HEADER\_NOT\_VALID | 2047 | 此头部域对该资源无效 |
| NETDEV\_E\_LIVE\_INVALID\_RANGE | 2048 | 无效范围 |
| NETDEV\_E\_LIVE\_PARAMETER\_READ\_ONLY | 2049 | 参数是只读的 |
| NETDEV\_E\_LIVE\_AO\_NOT\_ALLOWED | 2050 | 不允许合控制 |
| NETDEV\_E\_LIVE\_ONLY\_AO\_ALLOWED | 2051 | 只允许合控制 |
| NETDEV\_E\_LIVE\_UNSUPPORTED\_TRANSPORT | 2052 | 传输方式不支持 |
| NETDEV\_E\_LIVE\_DESTINATION\_UNREACHABLE | 2053 | 无法到达目的地址 |
| NETDEV\_E\_LIVE\_INTERNAL\_SERVER\_ERROR | 2054 | 服务器内部错误 |
| NETDEV\_E\_LIVE\_NOT\_IMPLEMENTED | 2055 | 未实现 |
| NETDEV\_E\_LIVE\_BAD\_GATEWAY | 2056 | 网关错误 |
| NETDEV\_E\_LIVE\_SERVICE\_UNAVAILABLE | 2057 | 无法得到服务 |
| NETDEV\_E\_LIVE\_VERSION\_NOT\_SUPPORTED | 2058 | 不支持此RTSP版本 |
| NETDEV\_E\_LIVE\_GATEWAY\_TIMEOUT | 2059 | 网关超时 |
| NETDEV\_E\_LIVE\_OPTION\_NOT\_SUPPORTED | 2060 | 不支持选项 |
| NETDEV\_E\_LIVE\_MALLOC\_FAIL | 2061 | 内存分配失败 |
| NETDEV\_E\_LIVE\_REALLOC\_FAIL | 2062 | 内存再分配失败 |
| NETDEV\_E\_LIVE\_DESCRIBE\_TIMEOUT | 2063 | describe超时 |
| NETDEV\_E\_LIVE\_IPC\_NOTBIND | 2064 | 通道未绑定 |
| NETDEV\_E\_LIVE\_DISK\_ABNOMAL | 2065 | 磁盘异常 |
| NETDEV\_E\_AUDIO\_EXISTED | 2100 | 语音对讲已存在 |
| NETDEV\_E\_AUDIO\_NO\_EXISTED | 2101 | 语音业务不存在 |
| NETDEV\_E\_AUDIO\_RESCODE\_INVALID | 2102 | 语音对讲资源编码无效 |
| NETDEV\_E\_AUDIO\_FAILED | 2104 | 语音对讲失败 |
| NETDEV\_E\_AUDIO\_AUDIOBCAST\_FULL | 2205 | 语音业务已满 |
| NETDEV\_E\_CAPTURE\_NO\_SUPPORT\_FORMAT | 2200 | 抓拍格式不支持 |
| NETDEV\_E\_CAPTURE\_NO\_ENOUGH\_CAPACITY | 2201 | 硬盘空间不足 |
| NETDEV\_E\_CAPTURE\_NO\_DECODED\_PICTURE | 2202 | 没有解码过的图片可供抓拍 |
| NETDEV\_E\_CAPTURE\_SINGLE\_FAILED | 2203 | 单次抓拍操作失败 |
| NETDEV\_E\_OVER\_ABILITY | 2301 | 码流超出能力集 |
| NETDEV\_E\_CLOUD\_DOWNLOAD\_FINISH | 2793 | 下载完成 |
| NETDEV\_E\_CLOUD\_PARSE\_DOMAIN\_FAIL | 2794 | 解析域名失败 |
| NETDEV\_E\_CLOUD\_CONNECT\_FAIL | 2795 | 连接失败 |
| NETDEV\_E\_CLOUD\_CONNECT\_TIMEOUT | 2796 | 连接超时 |
| NETDEV\_E\_CLOUD\_DOWNLOAD\_TIMEOUT | 2797 | 下载超时 |
| NETDEV\_E\_CLOUD\_DOWNLOAD\_FAIL | 2798 | 下载失败 |
| NETDEV\_E\_CLOUD\_NETWORK\_POOR | 2799 | 网络较差 |
| NETDEV\_E\_CLOUD\_PLAY\_FINISH | 2800 | 播放完成 |
| NETDEV\_E\_CLOUD\_DISK\_FULL | 2801 | 磁盘空间满 |
| NETDEV\_E\_CLOUD\_AUTH\_FAIL | 2802 | 鉴权失败 |
| NETDEV\_E\_CLOUD\_CURRENT\_TIME | 2803 | 当前播放时间，仅用于上报 |
| NETDEV\_E\_CLOUD\_PRIOR\_DISK\_FULL | 2804 | 磁盘预值满 |
| NETDEV\_E\_CLOUD\_NODE\_NOT\_EXIST | 2805 | 时间节点不存在 |
| NETDEV\_E\_CLOUD\_NO\_CACHE\_PATH | 2806 | 未设置缓存路径 |
| NETDEV\_E\_CLOUD\_MSG\_SEND\_FAIL | 2807 | 消息发送失败 |
| NETDEV\_E\_CLOUD\_TASK\_CANCELLED | 2808 | 任务已取消 |
| NETDEV\_E\_CLOUD\_TASK\_STREAM\_CONTINUE | 2809 | 流继续播放 |
| NETDEV\_E\_MEDIA\_INPUT\_NOT\_READY | 10000 | 媒体流未准备就绪 |
| NETDEV\_E\_CCB\_STATR\_INVALID | 10001 | 控制块状态不可用 |
| NETDEV\_E\_MEDIA\_OUTPUT\_BUSY | 10002 | 实况业务显示资源繁忙 |
| NETDEV\_E\_MEDIA\_START\_LOCAL\_LIVE\_ERR | 10003 | 实况媒体流未准备就绪 |
| NETDEV\_E\_MEDIA\_START\_LOCAL\_REPLAY\_ERR | 10004 | 回放媒体流未准备就绪 |
| NETDEV\_E\_MEDIA\_BW\_RECV\_NOT\_ENOUGH | 10007 | 网络接收带宽不足 |
| NETDEV\_E\_MEDIA\_BW\_SEND\_NOT\_ENOUGH | 10008 | 网络发送带宽不足 |
| NETDEV\_E\_MEDIA\_AUDIO\_BROADCAST\_TO\_LIMIT | 10009 | 语音广播业务已达上限 |
| NETDEV\_E\_MEDIA\_AUDIO\_CHL\_BING\_USED | 10010 | 音频通道已被占用 |
| NETDEV\_E\_MEDIA\_NOT\_SUPPORT\_ENCODETYPE | 10012 | 码流格式不支持 |
| NETDEV\_E\_MEDIA\_MAX | 10399 | 媒体相关错误码最大值 |

### 云台模块错误码

|  |  |  |
| --- | --- | --- |
| **错误码** | **错误码值** | **含义** |
| NETDEV\_E\_SET\_PRESET\_FAILED | 10400 | 预置位设置失败 |
| NETDEV\_E\_QUERY\_PRESET\_FAILED | 10401 | 预置位查询失败 |
| NETDEV\_E\_QUERY\_TRACK\_FAILED | 10402 | 轨迹查询失败 |
| NETDEV\_E\_START\_RECORD\_TRACK\_FAILED | 10403 | 开始录制轨迹失败 |
| NETDEV\_E\_STOP\_RECORD\_TRACK\_FAILED | 10404 | 结束录制轨迹失败 |
| NETDEV\_E\_QUERY\_CRUISE\_FAILED | 10405 | 巡航线路查询失败 |
| NETDEV\_E\_SET\_CRUISE\_FAILED | 10406 | 巡航线路设置失败 |
| NETDEV\_E\_PTZ\_COMMAND\_FAILED | 10407 | 云台操作失败 |
| NETDEV\_E\_PRESET\_IN\_CRUISE | 10408 | 预置位在巡航线路中使用无法删除 |
| NETDEV\_E\_CRUISEPOINT\_ER | 10409 | 设置巡航点不连续 |
| NETDEV\_E\_TRACK\_ISUSED | 10410 | 轨迹已使用无法删除 |
| NETDEV\_E\_SERIALMODE\_MISMATCH | 10411 | 串口模式不匹配 |
| NETDEV\_E\_TRACK\_NOT\_EXIST | 10412 | 轨迹不存在 |
| NETDEV\_E\_MODE\_CRUISE\_FULL | 10413 | 模式路径轨迹点个数已满 |
| NETDEV\_E\_PTZ\_ISUSED | 10414 | 云台使用中 |
| NETDEV\_E\_PRESET\_IN\_GUARD | 10415 | 预置位在守望中使用 |
| NETDEV\_E\_CRUISE\_PATH\_IN\_GUARD | 10416 | 巡航路径在守望中使用 |
| NETDEV\_E\_PTZ\_MAX | 10799 | 云台模块错误码最大值 |

### 回放模块错误码

|  |  |  |
| --- | --- | --- |
| **错误码** | **错误码值** | **含义** |
| NETDEV\_E\_VOD\_ABEND | 10800 | 回放异常结束 |
| NETDEV\_E\_VOD\_END | 10801 | 回放结束 |
| NETDEV\_E\_VOD\_CALL\_DRV | 10802 | 调用驱动接口失败 |
| NETDEV\_E\_VOD\_CB\_NOT\_FIND | 10803 | 控制块未找到 |
| NETDEV\_E\_VOD\_OVER\_ABILITY | 10804 | 超过回放能力 |
| NETDEV\_E\_VOD\_TAG\_OVER\_LIMIT | 10805 | 标签数量超过限制 |
| NETDEV\_E\_VOD\_NO\_RECORDING | 10806 | 无录像内容 |
| NETDEV\_E\_VOD\_SMART\_NOT\_SUPPORT | 10807 | 国标设备不支持智能回放 |
| NETDEV\_E\_VOD\_NO\_REPLAYURL | 10808 | 无法获取回放的url |
| NETDEV\_E\_VOD\_MAX | 10999 | 回放模块错误码最大值 |

### 存储模块错误码

|  |  |  |
| --- | --- | --- |
| **错误码** | **错误码值** | **含义** |
| NETDEV\_E\_STOR\_ABEND | 101000 | 存储异常结束 |
| NETDEV\_E\_ARRAY\_NAME\_EXITS | 11001 | 阵列已存在 |
| NETDEV\_E\_NOT\_RAID\_MODE | 11002 | 非阵列模式 |
| NETDEV\_E\_NO\_STORE\_PLAN | 11003 | 该通道没有配置过录像计划 |
| NETDEV\_E\_DISK\_ZERO\_OR\_ABNORMAL | 11022 | 无硬盘或硬盘异常 |
| NETDEV\_E\_SLOT\_NOT\_NORMAL | 11023 | 槽位1磁盘状态不正常 |
| NETDEV\_E\_SLOT\_RECONNECT | 11024 | 开启人脸后槽位1拔掉又插上需要重启 |
| NETDEV\_E\_BATTERY\_STATUS\_FAULT | 11034 | 设备电池状态异常 |
| NETDEV\_E\_STOR\_MAX | 11199 | 存储模块错误码最大值 |

### 用户管理模块错误码

|  |  |  |
| --- | --- | --- |
| **错误码** | **错误码值** | **含义** |
| NETDEV\_E\_USER\_NOT\_ONLINE | 101200 | 用户不在线 |
| NETDEV\_E\_USER\_WRONG\_PASSWD | 11201 | 用户密码错误 |
| NETDEV\_E\_USER\_NO\_SUCH\_USER | 11202 | 没有该用户 |
| NETDEV\_E\_USER\_NO\_AUTH | 11203 | 用户无权限 |
| NETDEV\_E\_USER\_MAX\_NUM | 11204 | 用户上限-不能再被添加 |
| NETDEV\_E\_USER\_EXIST | 11205 | 用户已存在 |
| NETDEV\_E\_USER\_LOGIN\_MAX\_NUM | 11206 | 用户登录数已达上限 |
| NETDEV\_E\_USER\_LOCKED | 11207 | 用户已被锁定 |
| NETDEV\_E\_USER\_IS\_ADMIN | 11208 | 不允许此操作，用户是系统管理员 |
| NETDEV\_E\_USER\_NOT\_VALID\_PERIOD | 11209 | 不在有效期内 |
| NETDEV\_E\_USER\_ROLE\_BEYOND\_RANGE | 11210 | 用户角色个数超过上限 |
| NETDEV\_E\_USER\_SELOG\_IP\_LOCKED | 11211 | 用户安全登录锁定 |
| NETDEV\_E\_ROLE\_EXIST | 11212 | 角色已存在 |
| NETDEV\_E\_USER\_NOT\_VALID\_TIME | 11213 | 不在有效时段内 |
| NETDEV\_E\_USER\_WEAK\_PASSWD\_LOGIN\_FAIL | 11214 | 登录失败,请在本地网络修改成强密码后再次尝试远程登录 |
| NETDEV\_E\_USER\_IS\_TEMPPASSWD | 11217 | 用户使用临时密码登录 |
| NETDEV\_E\_USER\_TEMPPASSWD\_LOGIN\_FAIL | 11218 | 公网临时密码禁止登陆 |
| NETDEV\_E\_USER\_WEAK\_PASSWD\_MODIFY | 11219 | 强制修改密码 |
| NETDEV\_E\_USER\_IP\_NO\_AUTH | 11221 | 登录IP无权限 |
| NETDEV\_E\_USER\_MAX | 11599 | 用户模块错误码最大值 |

### 设备管理模块错误码

|  |  |  |
| --- | --- | --- |
| **错误码** | **错误码值** | **含义** |
| NETDEV\_E\_DEV\_REPEAT\_DEV | 11602 | 重复设备 |
| NETDEV\_E\_DEV\_NO\_SUCH\_DEV | 11603 | 没有该设备 |
| NETDEV\_E\_DEV\_NO\_SUCH\_CHL | 11604 | 没有该通道 |
| NETDEV\_E\_DEV\_MAX\_NUM | 11605 | 一体机设备上限-不能再被添加 |
| NETDEV\_E\_DEV\_LOGIN\_MAX\_NUM | 11606 | 设备登录数上限 |
| NETDEV\_E\_DEV\_COMMON\_FAILED | 11607 | 由设备返回的通用错误 |
| NETDEV\_E\_DEV\_NO\_SUCH\_DEV\_OID | 11608 | 不存在设备历史ID |
| NETDEV\_E\_DEV\_DEL\_FORBID | 11609 | 设备不允许删除 |
| NETDEV\_E\_CHL\_OFFLINE | 11610 | 通道不在线 |
| NETDEV\_E\_DEV\_TYPE\_UNMATCHED | 11611 | 所添加设备类型不匹配 |
| NETDEV\_E\_DEV\_REPEAT\_DEV\_NAME | 11612 | 设备名称重复 |
| NETDEV\_E\_DEV\_CFG\_FAILED | 11613 | 设备配置失败 |
| NETDEV\_E\_DEV\_NO\_RESULT | 11614 | 查询无结果 |
| NETDEV\_E\_DEV\_DEC\_OFFLINE | 11617 | 解码设备离线 |
| NETDEV\_E\_DEV\_VCA\_ALARM\_NO\_PIC | 11661 | 未获取到VCA告警图片 |
| NETDEV\_E\_COMMON\_FAIL\_STAT | 11801 | 读取文件（目录）状态失败 |
| NETDEV\_E\_COMMON\_FILE\_NONEXIST | 11802 | 文件不存在 |
| NETDEV\_E\_COMMON\_MKDIR\_ERR | 11803 | 创建目录失败 |
| NETDEV\_E\_SUBSCRIBE\_FULL | 11804 | 本用户订阅已满,设备侧用户订阅已满的错误码不能返回 |
| NETDEV\_E\_UPGRADE\_NOTADMIN | 11805 | 非admin用户无法升级 |
| NETDEV\_E\_UPGRADE\_INVALID | 11806 | 升级未开始 |
| NETDEV\_E\_UPGRADE\_INPROCESS | 11807 | 升级处理中 |
| NETDEV\_E\_UPGRADE\_NOMEMORY | 11808 | 升级内存空间不够 |
| NETDEV\_E\_UPGRADE\_FILE\_OPEN\_ERR | 11809 | 升级打开镜像文件出 |
| NETDEV\_E\_UPGRADE\_DEVICE\_ERR | 11810 | 升级FLASH出错 |
| NETDEV\_E\_UPGRADE\_BUSY | 11811 | 不能同时加载多个升级进程 |
| NETDEV\_E\_UPGRADE\_FAIL\_TIMEOUT | 11812 | 升级超时 |
| NETDEV\_E\_INVALID\_CONFIGFILE | 11813 | 配置文件无效 |
| NETDEV\_E\_STOR\_RESOURCE\_NOTINIT | 11814 | 存储资源未分配 |
| NETDEV\_E\_DEV\_MAX | 11999 | 设备模块错误码最大值 |

### 组织管理模块错误码

|  |  |  |
| --- | --- | --- |
| **错误码** | **错误码值** | **含义** |
| NETDEV\_E\_ORG\_MAX\_NUM | 12000 | 组织上限-不能再被添加 |
| NETDEV\_E\_ORG\_MAX\_LEVEL | 12001 | 组织层级上限 |
| NETDEV\_E\_ORG\_RESOURCE\_EXIST | 12002 | 组织下有资源-不能被删除 |
| NETDEV\_E\_ORG\_EXIST | 12003 | 同分支下组织已存在 |
| NETDEV\_E\_ORG\_NO\_SUCH\_ORG | 12004 | 组织不存在 |
| NETDEV\_E\_ORG\_IS\_ROOT | 12005 | 不允许此操作，组织是根组织 |
| NETDEV\_E\_ORG\_NO\_SUCH\_PORG | 12006 | 父组织不存在 |
| NETDEV\_E\_ORG\_SUB\_EXIST | 12007 | 组织下有组织,不能被删除 |
| NETDEV\_E\_ORG\_MAX | 12399 | 组织模块最大错误码值 |

### 电视墙管理模块错误码

|  |  |  |
| --- | --- | --- |
| **错误码** | **错误码值** | **含义** |
| NETDEV\_E\_TVWALL\_WND\_NUMLIMITED | 12400 | 窗口数量超过限制 |
| NETDEV\_E\_TVWALL\_NUMLIMTED | 12401 | 电视墙数量超过限制 |
| NETDEV\_E\_TVWALL\_LED\_NUMLIMTED | 12402 | 虚拟LED数量超过限制 |
| NETDEV\_E\_TVWALL\_LED\_LINELIMTED | 12403 | 单个虚拟LED行数超出限制 |
| NETDEV\_E\_TVWALL\_SCENE\_OVER\_LIMIT | 12404 | 单电视墙场景个数超出限制 |
| NETDEV\_E\_TVWALL\_BASEPIC\_NUMLIMITED | 12405 | 设备能存储的底图数量超出限制 |
| NETDEV\_E\_TVWALL\_BASEPIC\_SCENELIMITED | 12406 | 单场景能绑定的底图数量超出限制 |
| NETDEV\_E\_TVWALL\_DECODE\_OUTRANGE | 12407 | 目的位置已超解码能力 |
| NETDEV\_E\_TVWALL\_WND\_OUTRANGE | 12408 | 开窗面积已超过三倍面积 |
| NETDEV\_E\_TVWALL\_WND\_RANGELIMITED | 12409 | 开窗位置/漫游位置超出范围 |
| NETDEV\_E\_TVWALL\_SCROLL\_LED\_NUMLIMITED | 12410 | 滚动虚拟LED超出限制 |
| NETDEV\_E\_TVWALL\_WND\_LOCKED | 12430 | 当前窗口已被锁定 |
| NETDEV\_E\_TVWALL\_LED\_NOTEXIST | 12431 | 虚拟LED不存在 |
| NETDEV\_E\_TVWALL\_SEQ\_EXISTD | 12432 | 轮切已经存在 |
| NETDEV\_E\_TVWALL\_WND\_NOTEXIST | 12433 | 窗口不存在 |
| NETDEV\_E\_TVWALL\_WND\_EXIST | 12434 | 窗口已经存在 |
| NETDEV\_E\_TVWALL\_SPILT\_NOTEXIST | 12435 | 窗口中的分屏不存在 |
| NETDEV\_E\_TVWALL\_WND\_NOSPILT | 12436 | 窗口不能分屏 |
| NETDEV\_E\_TVWALL\_CHANNEL\_UESED | 12437 | 解码通道被其他电视墙占用 |
| NETDEV\_E\_TVWALL\_WINDEXIST | 12438 | 电视墙存在窗口 |
| NETDEV\_E\_TVWALL\_NOTEXIST | 12439 | 电视墙信息不存在 |
| NETDEV\_E\_TVWALL\_SPLIT\_INFOERR | 12440 | 错误的分屏类型 |
| NETDEV\_E\_TVWALL\_CHANNEL\_NUMERR | 12411 | 通道数量错误 |
| NETDEV\_E\_TVWALL\_ENC\_ERR | 12442 | 媒体流未准备就绪 |
| NETDEV\_E\_TVWALL\_PERSCREEN\_NUMLIMITED | 12443 | 屏幕（单VO口）窗口数量超过限制 |
| NETDEV\_E\_TVWALL\_MAX | 12799 | 电视墙模块最大错误码值 |

### 网络相关错误码

|  |  |  |
| --- | --- | --- |
| **错误码** | **错误码值** | **含义** |
| NETDEV\_E\_IPADDR\_CONFLICT | 12800 | IP地址冲突 |
| NETDEV\_E\_UPNP\_PORT\_DISCARD | 12801 | 映射的端口已废弃 |
| NETDEV\_E\_UPNP\_PORT\_EMPLOY | 12802 | 端口已占用 |
| NETDEV\_E\_SET\_VMP\_PORT\_FAILED | 12803 | 设置VMP端口失败 |
| NETDEV\_E\_SET\_MEDIA\_PORT\_FAILED | 12804 | 设置媒体流端口失败 |
| NETDEV\_E\_SET\_RTSP\_PORT\_FAILED | 12805 | 设置RTSP端口失败 |
| NETDEV\_E\_SET\_ONVIF\_PORT\_FAILED | 12806 | 设置ONVIF端口失败 |
| NETDEV\_E\_SET\_HTTPS\_PORT\_FAILED | 12807 | 设置HTTPS端口失败 |
| NETDEV\_E\_SET\_HTTP\_PORT\_FAILED | 12808 | 设置HTTP端口失败 |
| NETDEV\_E\_TEST\_EMAIL | 12809 | 发送测试邮件失败 |
| NETDEV\_E\_DDNS\_INIT | 12810 | 初始化默认值 |
| NETDEV\_E\_DDNS\_GOOD | 12811 | 更新完成 |
| NETDEV\_E\_DDNS\_NOCHG | 12812 | 更新完成,但是ip地址没有变化 |
| NETDEV\_E\_DDNS\_BADAUTH | 12813 | 错误的用户名密码 |
| NETDEV\_E\_DDNS\_DONATOR | 12814 | 专用的选项要求类似offline |
| NETDEV\_E\_DDNS\_NOTFQ | 12815 | 主机名格式不合格 |
| NETDEV\_E\_DDNS\_NOHOST | 12816 | 不存在该主机名账户 |
| NETDEV\_E\_DDNS\_NUMHOST | 12817 | 请求立即更新多于20个主机 |
| NETDEV\_E\_DDNS\_ABUSE | 12818 | 主机名由于滥用而被禁用 |
| NETDEV\_E\_DDNS\_BADAGENT | 12819 | 代理发送错误的HTTP请求格式 |
| NETDEV\_E\_DDNS\_GOOD\_127 | 12820 | 除非客户端想要更新127.0.0.1，否则该信息表明在http请求规格中发生错误 |
| NETDEV\_E\_DDNS\_DNSERR | 12821 | DNS错误，请联系Support |
| NETDEV\_E\_DDNS\_911 | 12822 | 服务维护正在处理 |
| NETDEV\_E\_DDNS\_REQ | 12823 | DDNS请求发送失败 |
| NETDEV\_E\_DDNS\_CHECK\_KEY | 12824 | 校验码错误 |
| NETDEV\_E\_DDNS\_BAD\_NAME\_FORMAT | 12825 | 域名格式不合格 |
| NETDEV\_E\_DDNS\_DOMAIN\_CONFLICT | 12826 | 域名冲突 |
| NETDEV\_E\_DDNS\_CLOSE | 12827 | 非ddns服务器返回，只用于关闭ddns记录状态 |
| NETDEV\_E\_CREATE\_CERT\_FAILD | 12828 | 证书文件生成失败 |
| NETDEV\_E\_GET\_CERT\_FAILD | 12829 | 上传证书文件不合法 |
| NETDEV\_E\_TEST\_EMAIL\_CHECKSERVER\_INFO | 12830 | 认证服务器用户名或密码错误 |
| NETDEV\_E\_TEST\_EMAIL\_COMM | 12831 | 邮件发送失败,请检查网络和邮件配置 |
| NETDEV\_E\_TEST\_EMAIL\_TIMEOUT | 12832 | 邮件发送超时 |
| NETDEV\_E\_CLOUD\_REPEAT\_USER | 12833 | 该云端账号已登录 |
| NETDEV\_E\_CLOUD\_DEV\_ADD\_FAIL | 12834 | 添加云端设备信息失败 |
| NETDEV\_E\_CLOUD\_ORG\_NULL | 12835 | 云端组织信息为空 |
| NETDEV\_E\_GET\_CLOUD\_ORG | 12836 | 无法获取云端信息，请检查网络配置 |
| NETDEV\_E\_NO\_SUCH\_CLOUD\_USER | 12837 | 云端账号不存在 |
| NETDEV\_E\_NET\_MAX | 13199 | 网络模块最大错误码值 |

# 产品相关接口定义

## 通用接口

|  |  |
| --- | --- |
| **功能** | **接口** |
| SDK初始化 | NETDEV\_Init |
| SDK清理 | NETDEV\_Cleanup |
| 设置日志大小和数量 | NETDEV\_ConfigLogFile |
| 设置日志路径 | NETDEV\_SetLogPath |
| 设置写入日志路径 | NETDEV\_SetWriteLogFlag |
| 获取SDK版本信息 | NETDEV\_GetSDKVersion |
| 获取接口错误码 | NETDEV\_GetLastError |
| 注册异常回调函数 | NETDEV\_SetExceptionCallBack |
| 设置最大解码路数 | NETDEV\_SetMaxDecChlNum |
| 设置保活等待间隔及次数 | NETDEV\_SetConnectTime |
| 设置接收超时时间 | NETDEV\_SetRevTimeOut |
| 设置设备搜索回调函数 | NETDEV\_SetDiscoveryCallBack |
| 设备搜索 | NETDEV\_Discovery |
| 设备登录 | NETDEV\_Login\_V30 |
| 设备注销 | NETDEV\_Logout |
| 动态产生一个关键帧 | NETDEV\_MakeKeyFrame |
| 实时预览 | NETDEV\_RealPlay |
| 停止实时预览 | NETDEV\_StopRealPlay |
| 启动本地录像 | NETDEV\_SaveRealData |
| 停止本地录像 | NETDEV\_StopSaveRealData |
| 获取实况起流URL | NETDEV\_GetStreamUrl |
| 通过URL启动实况预览 | NETDEV\_FastRealPlayByUrl |
| 查找录像文件列表 | NETDEV\_FindFile |
| 逐个获取查找到的文件信息 | NETDEV\_FindNextFile |
| 关闭录像文件查找 | NETDEV\_FindClose |
| LAPI告警订阅 | NETDEV\_SubscibeLapiAlarm |
| 取消LAPI告警订阅 | NETDEV\_UnSubLapiAlarm |
| 注册码流回调函数:原始码流 | NETDEV\_SetPlayDataCallBack |
| 注册实况码流回调函数:拼帧后码流数据 | NETDEV\_SetPlayParseCB |
| 注册实时码流回调函数：解码后视频媒体流数据 | NETDEV\_SetPlayDecodeVideoCB |
| 注册码流回调函数：拼帧后音频媒体流数据 | NETDEV\_SetPlayParseAudioCB |
| 注册码流回调函数：解码后音频媒体流数据 | NETDEV\_SetPlayDecodeAudioCB |
| 设置显示后数据回调 | NETDEV\_SetPlayDisplayCB |
| 获取窗口码率 | NETDEV\_GetBitRate |
| 获取窗口帧率 | NETDEV\_GetFrameRate |
| 获取窗口编码格式 | NETDEV\_GetVideoEncodeFmt |
| 获取视频分辨率 | NETDEV\_GetResolution |
| 获取窗口丢包率 | NETDEV\_GetLostPacketRate |
| 重置窗口丢包率 | NETDEV\_ResetLostPacketRate |
| 设置视频图像显示比例 | NETDEV\_SetRenderScale |
| 影像参数获取,只获取当前画面参数 | NETDEV\_GetVideoEffect |
| 影像调节,只改变当前画面 | NETDEV\_SetVideoEffect |
| 设置数字放大 | NETDEV\_SetDigitalZoom |
| 实况抓拍 | NETDEV\_CapturePicture |
| 设置流畅性优先 | NETDEV\_SetPictureFluency |
| 开启语音对讲 | NETDEV\_StartVoiceCom |
| 关闭语音对讲 | NETDEV\_StopVoiceCom |
| 开启麦克风 | NETDEV\_OpenMic |
| 关闭麦克风 | NETDEV\_CloseMic |
| 调节麦克风音量 | NETDEV\_MicVolumeControl |
| 获取麦克风音量 | NETDEV\_GetMicVolume |
| 获取静音状态 | NETDEV\_GetMuteStatus |
| 设置静音状态 | NETDEV\_SetMuteStatus |
| 开启声音 | NETDEV\_OpenSound |
| 调节扬声器音量 | NETDEV\_SoundVolumeControl |
| 获取扬声器音量 | NETDEV\_GetSoundVolume |
| 关闭声音 | NETDEV\_CloseSound |
| 开启输入语音数据服务 | NETDEV\_StartInputVoiceSrv |
| 关闭输入语音数据服务 | NETDEV\_StopInputVoiceSrv |
| 输入语音数据 | NETDEV\_InputVoiceData |
| 设置解析后语音数据回调函数 | NETDEV\_SetParseVoiceDataCB |
| 按时间回放录像文件 | NETDEV\_PlayBackByTime |
| 控制录像回放的状态 | NETDEV\_PlayBackControl |
| 停止回放业务 | NETDEV\_StopPlayBack |
| 按时间下载录像文件 | NETDEV\_GetFileByTime |
| 停止下载录像文件 | NETDEV\_StopGetFile |
| 打开本地文件 | NETDEV\_OpenMediaFile |
| 播放本地文件 | NETDEV\_StartPlayMediaFile |
| 停止播放本地文件业务 | NETDEV\_StopPlayMediaFile |
| 获取媒体文件时间总长度 | NETDEV\_GetMediaFileTime |
| 云台控制操作(不启动实况预览) | NETDEV\_PTZControl\_Other |
| 获取云台预置位列表 | NETDEV\_GetPTZPresetList |
| 云台预置位操作(不需要启动预览) | NETDEV\_PTZPreset\_Other |
| 获取云台预置位巡航路径 | NETDEV\_PTZGetCruise |
| 云台预置位巡航操作(不需要启动预览) | NETDEV\_PTZCruise\_Other |
| 获取云台轨迹巡航路径 | NETDEV\_PTZGetTrackCruise |
| 云台轨迹巡航操作(不需要启动预览) | NETDEV\_PTZTrackCruise |
| 拉框放大.缩小(不需要启动预览) | NETDEV\_PTZSelZoomIn\_Other |
| 云台3D定位(不需要启动预览) | NETDEV\_PTZ3DPosition\_Other |
| 创建人员库信息 | NETDEV\_CreatePersonLibInfo |
| 修改人员库信息 | NETDEV\_ModifyPersonLibInfo |
| 查询所有已创建的人员库信息 | NETDEV\_FindPersonLibList |
| 逐个获取查找到的 人脸库 信息 | NETDEV\_FindNextPersonLibInfo |
| 关闭查找 人脸库，释放资源 | NETDEV\_FindClosePersonLibList |
| 删除指定的人员库 | NETDEV\_DeletePersonLibInfo |
| 新增指定的人员信息 | NETDEV\_AddPersonInfo |
| 修改指定的人员信息 | NETDEV\_ModifyPersonInfo |
| 删除指定的人员信息 | NETDEV\_DeletePersonInfo |
| 查询所有人脸布控任务 | NETDEV\_FindPersonMonitorList |
| 逐个获取查找到的 布控任务 信息 | NETDEV\_FindNextPersonMonitorInfo |
| 关闭查找 布控任务，释放资源 | NETDEV\_FindClosePersonMonitorList |
| 新增单个人脸布控任务 | NETDEV\_AddPersonMonitorInfo |
| 删除单个人脸布控任务 | NETDEV\_DeletePersonMonitorInfo |
| 查询单个人脸布控任务配置信息 | NETDEV\_GetPersonMonitorRuleInfo |
| 设置单个人脸布控任务配置信息 | NETDEV\_SetPersonMonitorRuleInfo |
| 批量删除人脸布控任务 | NETDEV\_BatchDeletePersonMonitorInfo |
| 查询所有人员库的容量信息 | NETDEV\_GetPersonLibCapacity |
| 条件查询人员信息 | NETDEV\_FindPersonInfoList |
| 逐个获取查找到的 人员 信息 | NETDEV\_FindNextPersonInfo |
| 关闭查找 人员信息，释放资源 | NETDEV\_FindClosePersonInfoList |
| 注册人脸报警消息回调函数 | NETDEV\_SetPersonAlarmCallBack |
| 注册结构化报警消息回调函数 | NETDEV\_SetStructAlarmCallBack |

## 网络摄像机

|  |  |
| --- | --- |
| **功能** | **接口** |
| 查询视频通道信息列表 | NETDEV\_QueryVideoChlDetailList |
| 查询视频通道信息列表 | NETDEV\_QueryVideoChlDetailListEx |
| 启动实时预览 | NETDEV\_RealPlay |
| 停止实时预览 | NETDEV\_StopRealPlay |
| 注册码流回调函数:原始码流 | NETDEV\_SetPlayDataCallBack |
| 注册实况码流回调函数:拼帧后码流数据 | NETDEV\_SetPlayParseCB |
| 注册实时码流回调函数：解码后视频媒体流数据 | NETDEV\_SetPlayDecodeVideoCB |
| 注册码流回调函数：拼帧后音频媒体流数据 | NETDEV\_SetPlayParseAudioCB |
| 注册码流回调函数：解码后音频媒体流数据 | NETDEV\_SetPlayDecodeAudioCB |
| 设置显示后数据回调 | NETDEV\_SetPlayDisplayCB |
| 获取窗口码率 | NETDEV\_GetBitRate |
| 获取窗口帧率 | NETDEV\_GetFrameRate |
| 获取窗口编码格式 | NETDEV\_GetVideoEncodeFmt |
| 获取视频分辨率 | NETDEV\_GetResolution |
| 获取窗口丢包率 | NETDEV\_GetLostPacketRate |
| 重置窗口丢包率 | NETDEV\_ResetLostPacketRate |
| 设置视频图像显示比例 | NETDEV\_SetRenderScale |
| 影像参数获取,只获取当前画面参数 | NETDEV\_GetVideoEffect |
| 影像调节,只改变当前画面 | NETDEV\_SetVideoEffect |
| 设置数字放大 | NETDEV\_SetDigitalZoom |
| 实况抓拍 | NETDEV\_CapturePicture |
| 非预览下抓拍 | NETDEV\_CaptureNoPreview |
| 动态产生一个关键帧 | NETDEV\_MakeKeyFrame |
| 本地录像 | NETDEV\_SaveRealData |
| 停止本地录像 | NETDEV\_StopSaveRealData |
| 设置流畅性优先 | NETDEV\_SetPictureFluency |
| 开启语音对讲 | NETDEV\_StartVoiceCom |
| 关闭语音对讲 | NETDEV\_StopVoiceCom |
| 开启麦克风 | NETDEV\_OpenMic |
| 关闭麦克风 | NETDEV\_CloseMic |
| 调节麦克风音量 | NETDEV\_MicVolumeControl |
| 获取麦克风音量 | NETDEV\_GetMicVolume |
| 获取静音状态 | NETDEV\_GetMuteStatus |
| 设置静音状态 | NETDEV\_SetMuteStatus |
| 开启声音 | NETDEV\_OpenSound |
| 调节扬声器音量 | NETDEV\_SoundVolumeControl |
| 获取扬声器音量 | NETDEV\_GetSoundVolume |
| 关闭声音 | NETDEV\_CloseSound |
| 开启输入语音数据服务 | NETDEV\_StartInputVoiceSrv |
| 关闭输入语音数据服务 | NETDEV\_StopInputVoiceSrv |
| 输入语音数据 | NETDEV\_InputVoiceData |
| 设置解析后语音数据回调函数 | NETDEV\_SetParseVoiceDataCB |
| 根据文件类型.时间查找设备录像文件 | NETDEV\_FindFile |
| 逐个获取查找到的文件信息 | NETDEV\_FindNextFile |
| 关闭文件查找,释放资源 | NETDEV\_FindClose |
| 按名称回放录像文件 | NETDEV\_PlayBackByName |
| 按时间回放录像文件 | NETDEV\_PlayBackByTime |
| 控制录像回放的状态 | NETDEV\_PlayBackControl |
| 停止回放业务 | NETDEV\_StopPlayBack |
| 按文件名下载录像文件 | NETDEV\_GetFileByName |
| 按时间下载录像文件 | NETDEV\_GetFileByTime |
| 停止下载录像文件 | NETDEV\_StopGetFile |
| 打开本地文件 | NETDEV\_OpenMediaFile |
| 播放本地文件 | NETDEV\_StartPlayMediaFile |
| 停止播放本地文件业务 | NETDEV\_StopPlayMediaFile |
| 获取媒体文件时间总长度 | NETDEV\_GetMediaFileTime |
| 云台控制操作(不启动实况预览) | NETDEV\_PTZControl\_Other |
| 获取云台预置位列表 | NETDEV\_GetPTZPresetList |
| 云台预置位操作(不需要启动预览) | NETDEV\_PTZPreset\_Other |
| 获取云台预置位巡航路径 | NETDEV\_PTZGetCruise |
| 云台预置位巡航操作(不需要启动预览) | NETDEV\_PTZCruise\_Other |
| 获取云台轨迹巡航路径 | NETDEV\_PTZGetTrackCruise |
| 云台轨迹巡航操作(不需要启动预览) | NETDEV\_PTZTrackCruise |
| 拉框放大.缩小(不需要启动预览) | NETDEV\_PTZSelZoomIn\_Other |
| 云台3D定位(不需要启动预览) | NETDEV\_PTZ3DPosition\_Other |
| 获取设备的配置信息 | NETDEV\_GetDevConfig |
| 设置设备的配置信息 | NETDEV\_SetDevConfig |
| 修改设备名称 | NETDEV\_ModifyDeviceName |
| 获取映射端口 | NETDEV\_GetUpnpNatState |
| 设置映射端口 | NETDEV\_SetUpnpNatState |
| 获取设备系统时间配置 | NETDEV\_GetSystemTimeCfg |
| 设置设备系统时间配置 | NETDEV\_SetSystemTimeCfg |
| 重启设备 | NETDEV\_Reboot |
| 恢复出厂设置 | NETDEV\_RestoreConfig |
| 开启手动录像 | NETDEV\_StartManualRecord |
| 停止手动录像 | NETDEV\_StopManualRecord |
| 根据时间查询设备告警 | NETDEV\_FindAlarmInfoList |
| 逐个获取查找到的告警信息 | NETDEV\_FindNextAlarmInfo |
| 关闭告警查询，释放资源 | NETDEV\_FindCloseAlarmInfo |
| 获取所有用户全部信息 | NETDEV\_GetUserDetailList |
| 创建用户信息 | NETDEV\_CreateUser |
| 修改用户信息 | NETDEV\_ModifyUser |
| 删除用户信息 | NETDEV\_DeleteUser |
| 注册客流量统计上报回调函数 | NETDEV\_SetPassengerFlowStatisticCallBack |
| 注册人脸抓拍上报回调函数 | NETDEV\_SetFaceSnapshotCallBack |
| 创建人员库信息 | NETDEV\_CreatePersonLibInfo |
| 修改人员库信息 | NETDEV\_ModifyPersonLibInfo |
| 查询所有已创建的人员库信息 | NETDEV\_FindPersonLibList |
| 逐个获取查找到的 人脸库 信息 | NETDEV\_FindNextPersonLibInfo |
| 关闭查找 人脸库，释放资源 | NETDEV\_FindClosePersonLibList |
| 删除指定的人员库 | NETDEV\_DeletePersonLibInfo |
| 新增指定的人员信息 | NETDEV\_AddPersonInfo |
| 修改指定的人员信息 | NETDEV\_ModifyPersonInfo |
| 删除指定的人员信息 | NETDEV\_DeletePersonInfo |
| 查询所有人脸布控任务 | NETDEV\_FindPersonMonitorList |
| 逐个获取查找到的 布控任务 信息 | NETDEV\_FindNextPersonMonitorInfo |
| 关闭查找 布控任务，释放资源 | NETDEV\_FindClosePersonMonitorList |
| 新增单个人脸布控任务 | NETDEV\_AddPersonMonitorInfo |
| 删除单个人脸布控任务 | NETDEV\_DeletePersonMonitorInfo |
| 查询单个人脸布控任务配置信息 | NETDEV\_GetPersonMonitorRuleInfo |
| 设置单个人脸布控任务配置信息 | NETDEV\_SetPersonMonitorRuleInfo |
| 批量删除人脸布控任务 | NETDEV\_BatchDeletePersonMonitorInfo |
| 查询所有人员库的容量信息 | NETDEV\_GetPersonLibCapacity |
| 条件查询人员信息 | NETDEV\_FindPersonInfoList |
| 逐个获取查找到的 人员 信息 | NETDEV\_FindNextPersonInfo |
| 关闭查找 人员信息，释放资源 | NETDEV\_FindClosePersonInfoList |
| 注册人脸报警消息回调函数 | NETDEV\_SetPersonAlarmCallBack |
| 注册结构化报警消息回调函数 | NETDEV\_SetStructAlarmCallBack |
| LAPI告警订阅 | NETDEV\_SubscibeLapiAlarm |
| 取消LAPI告警订阅 | NETDEV\_UnSubLapiAlarm |

## 网络硬盘录像机

|  |  |
| --- | --- |
| **功能** | **接口** |
| 查询视频通道信息列表 | NETDEV\_QueryVideoChlDetailList |
| 查询视频通道信息列表 | NETDEV\_QueryVideoChlDetailListEx |
| 启动实时预览 | NETDEV\_RealPlay |
| 停止实时预览 | NETDEV\_StopRealPlay |
| 注册码流回调函数:原始码流 | NETDEV\_SetPlayDataCallBack |
| 注册实况码流回调函数:拼帧后码流数据 | NETDEV\_SetPlayParseCB |
| 注册实时码流回调函数：解码后视频媒体流数据 | NETDEV\_SetPlayDecodeVideoCB |
| 注册码流回调函数：拼帧后音频媒体流数据 | NETDEV\_SetPlayParseAudioCB |
| 注册码流回调函数：解码后音频媒体流数据 | NETDEV\_SetPlayDecodeAudioCB |
| 设置显示后数据回调 | NETDEV\_SetPlayDisplayCB |
| 获取窗口码率 | NETDEV\_GetBitRate |
| 获取窗口帧率 | NETDEV\_GetFrameRate |
| 获取窗口编码格式 | NETDEV\_GetVideoEncodeFmt |
| 获取视频分辨率 | NETDEV\_GetResolution |
| 获取窗口丢包率 | NETDEV\_GetLostPacketRate |
| 重置窗口丢包率 | NETDEV\_ResetLostPacketRate |
| 设置视频图像显示比例 | NETDEV\_SetRenderScale |
| 影像参数获取,只获取当前画面参数 | NETDEV\_GetVideoEffect |
| 影像调节,只改变当前画面 | NETDEV\_SetVideoEffect |
| 设置数字放大 | NETDEV\_SetDigitalZoom |
| 设置流畅性优先 | NETDEV\_SetPictureFluency |
| 开启语音对讲 | NETDEV\_StartVoiceCom |
| 关闭语音对讲 | NETDEV\_StopVoiceCom |
| 开启麦克风 | NETDEV\_OpenMic |
| 关闭麦克风 | NETDEV\_CloseMic |
| 调节麦克风音量 | NETDEV\_MicVolumeControl |
| 获取麦克风音量 | NETDEV\_GetMicVolume |
| 获取静音状态 | NETDEV\_GetMuteStatus |
| 设置静音状态 | NETDEV\_SetMuteStatus |
| 开启声音 | NETDEV\_OpenSound |
| 调节扬声器音量 | NETDEV\_SoundVolumeControl |
| 获取扬声器音量 | NETDEV\_GetSoundVolume |
| 关闭声音 | NETDEV\_CloseSound |
| 开启输入语音数据服务 | NETDEV\_StartInputVoiceSrv |
| 关闭输入语音数据服务 | NETDEV\_StopInputVoiceSrv |
| 输入语音数据 | NETDEV\_InputVoiceData |
| 设置解析后语音数据回调函数 | NETDEV\_SetParseVoiceDataCB |
| 实况抓拍 | NETDEV\_CapturePicture |
| 非预览下抓拍 | NETDEV\_CaptureNoPreview |
| 动态产生一个关键帧 | NETDEV\_MakeKeyFrame |
| 本地录像 | NETDEV\_SaveRealData |
| 停止本地录像 | NETDEV\_StopSaveRealData |
| 根据文件类型.时间查找设备录像文件 | NETDEV\_FindFile |
| 逐个获取查找到的文件信息 | NETDEV\_FindNextFile |
| 关闭文件查找,释放资源 | NETDEV\_FindClose |
| 按名称回放录像文件 | NETDEV\_PlayBackByName |
| 按时间回放录像文件 | NETDEV\_PlayBackByTime |
| 控制录像回放的状态 | NETDEV\_PlayBackControl |
| 停止回放业务 | NETDEV\_StopPlayBack |
| 按文件名下载录像文件 | NETDEV\_GetFileByName |
| 按时间下载录像文件 | NETDEV\_GetFileByTime |
| 停止下载录像文件 | NETDEV\_StopGetFile |
| 打开本地文件 | NETDEV\_OpenMediaFile |
| 播放本地文件 | NETDEV\_StartPlayMediaFile |
| 停止播放本地文件业务 | NETDEV\_StopPlayMediaFile |
| 获取媒体文件时间总长度 | NETDEV\_GetMediaFileTime |
| 云台控制操作(不启动实况预览) | NETDEV\_PTZControl\_Other |
| 获取云台预置位列表 | NETDEV\_GetPTZPresetList |
| 云台预置位操作(不需要启动预览) | NETDEV\_PTZPreset\_Other |
| 获取云台预置位巡航路径 | NETDEV\_PTZGetCruise |
| 云台预置位巡航操作(不需要启动预览) | NETDEV\_PTZCruise\_Other |
| 获取云台轨迹巡航路径 | NETDEV\_PTZGetTrackCruise |
| 云台轨迹巡航操作(不需要启动预览) | NETDEV\_PTZTrackCruise |
| 拉框放大.缩小(不需要启动预览) | NETDEV\_PTZSelZoomIn\_Other |
| 云台3D定位(不需要启动预览) | NETDEV\_PTZ3DPosition\_Other |
| 获取设备的配置信息 | NETDEV\_GetDevConfig |
| 设置设备的配置信息 | NETDEV\_SetDevConfig |
| 修改设备名称 | NETDEV\_ModifyDeviceName |
| 获取映射端口 | NETDEV\_GetUpnpNatState |
| 设置映射端口 | NETDEV\_SetUpnpNatState |
| 获取设备系统时间配置 | NETDEV\_GetSystemTimeCfg |
| 设置设备系统时间配置 | NETDEV\_SetSystemTimeCfg |
| 重启设备 | NETDEV\_Reboot |
| 恢复出厂设置 | NETDEV\_RestoreConfig |
| 开启手动录像 | NETDEV\_StartManualRecord |
| 停止手动录像 | NETDEV\_StopManualRecord |
| 获取所有用户全部信息 | NETDEV\_GetUserDetailList |
| 创建用户信息 | NETDEV\_CreateUser |
| 修改用户信息 | NETDEV\_ModifyUser |
| 删除用户信息 | NETDEV\_DeleteUser |
| 根据日志类型、时间查找日志信息 | NETDEV\_FindLogInfoList |
| 逐个获取查找到的日志信息 | NETDEV\_FindNextLogInfo |
| 关闭日志查找，释放资源 | NETDEV\_FindCloseLogInfo |
| 获取客流量统计信息列表 | NETDEV\_FindTrafficStatisticInfoList |
| 逐个查询获取到的客流量信息 | NETDEV\_FindNextTrafficStatisticInfo |
| 关闭查找，释放资源 | NETDEV\_FindCloseTrafficStatisticInfo |
| 获取客流量统计进度 | NETDEV\_GetTrafficStatisticProgress |
| 多通道开始客流量统计查询 | NETDEV\_StartMultiTrafficStatistic |
| 停止客流量查询 | NETDEV\_StopTrafficStatistic |
| 创建人员库信息 | NETDEV\_CreatePersonLibInfo |
| 修改人员库信息 | NETDEV\_ModifyPersonLibInfo |
| 查询所有已创建的人员库信息 | NETDEV\_FindPersonLibList |
| 逐个获取查找到的 人脸库 信息 | NETDEV\_FindNextPersonLibInfo |
| 关闭查找 人脸库，释放资源 | NETDEV\_FindClosePersonLibList |
| 删除指定的人员库 | NETDEV\_DeletePersonLibInfo |
| 新增指定的人员信息 | NETDEV\_AddPersonInfo |
| 修改指定的人员信息 | NETDEV\_ModifyPersonInfo |
| 删除指定的人员信息 | NETDEV\_DeletePersonInfo |
| 查询所有人脸布控任务 | NETDEV\_FindPersonMonitorList |
| 逐个获取查找到的 布控任务 信息 | NETDEV\_FindNextPersonMonitorInfo |
| 关闭查找 布控任务，释放资源 | NETDEV\_FindClosePersonMonitorList |
| 新增单个人脸布控任务 | NETDEV\_AddPersonMonitorInfo |
| 删除单个人脸布控任务 | NETDEV\_DeletePersonMonitorInfo |
| 批量删除人脸布控任务 | NETDEV\_BatchDeletePersonMonitorInfo |
| 查询单个人脸布控任务配置信息 | NETDEV\_GetPersonMonitorRuleInfo |
| 设置单个人脸布控任务配置信息 | NETDEV\_SetPersonMonitorRuleInfo |
| 查询所有人员库的容量信息 | NETDEV\_GetPersonLibCapacity |
| 条件查询人员信息 | NETDEV\_FindPersonInfoList |
| 逐个获取查找到的 人员 信息 | NETDEV\_FindNextPersonInfo |
| 关闭查找 人员信息，释放资源 | NETDEV\_FindClosePersonInfoList |
| 注册人脸报警消息回调函数 | NETDEV\_SetPersonAlarmCallBack |
| 注册结构化报警消息回调函数 | NETDEV\_SetStructAlarmCallBack |
| LAPI告警订阅 | NETDEV\_SubscibeLapiAlarm |
| 取消LAPI告警订阅 | NETDEV\_UnSubLapiAlarm |

## 一体机

|  |  |
| --- | --- |
| **功能** | **接口** |
| 获取指定组织信息 | NETDEV\_GetOrgInfo |
| 批量添加设备 | NETDEV\_BatchAddDevice |
| 批量删除设备 | NETDEV\_BatchDelDevice |
| 通过 设备类型 查询 设备列表 | NETDEV\_FindDevList |
| 逐个获取查找到的 设备信息 | NETDEV\_FindNextDevInfo |
| 关闭查找 设备信息，释放资源 | NETDEV\_FindCloseDevInfo |
| 通过设备ID或通道类型 查询通道信息列表 | NETDEV\_FindDevChnList |
| 逐个获取查找到的 设备通道 信息 | NETDEV\_FindNextDevChn |
| 关闭查找 设备通道信息，释放资源 | NETDEV\_FindCloseDevChn |
| 启动实时预览 | NETDEV\_RealPlay |
| 停止实时预览 | NETDEV\_StopRealPlay |
| 注册码流回调函数:原始码流 | NETDEV\_SetPlayDataCallBack |
| 注册实况码流回调函数:拼帧后码流数据 | NETDEV\_SetPlayParseCB |
| 注册实时码流回调函数：解码后视频媒体流数据 | NETDEV\_SetPlayDecodeVideoCB |
| 注册码流回调函数：拼帧后音频媒体流数据 | NETDEV\_SetPlayParseAudioCB |
| 注册码流回调函数：解码后音频媒体流数据 | NETDEV\_SetPlayDecodeAudioCB |
| 设置显示后数据回调 | NETDEV\_SetPlayDisplayCB |
| 获取窗口码率 | NETDEV\_GetBitRate |
| 获取窗口帧率 | NETDEV\_GetFrameRate |
| 获取窗口编码格式 | NETDEV\_GetVideoEncodeFmt |
| 获取视频分辨率 | NETDEV\_GetResolution |
| 获取窗口丢包率 | NETDEV\_GetLostPacketRate |
| 重置窗口丢包率 | NETDEV\_ResetLostPacketRate |
| 设置视频图像显示比例 | NETDEV\_SetRenderScale |
| 影像参数获取,只获取当前画面参数 | NETDEV\_GetVideoEffect |
| 影像调节,只改变当前画面 | NETDEV\_SetVideoEffect |
| 设置数字放大 | NETDEV\_SetDigitalZoom |
| 设置流畅性优先 | NETDEV\_SetPictureFluency |
| 开启语音对讲 | NETDEV\_StartVoiceCom |
| 关闭语音对讲 | NETDEV\_StopVoiceCom |
| 开启麦克风 | NETDEV\_OpenMic |
| 关闭麦克风 | NETDEV\_CloseMic |
| 调节麦克风音量 | NETDEV\_MicVolumeControl |
| 获取麦克风音量 | NETDEV\_GetMicVolume |
| 获取静音状态 | NETDEV\_GetMuteStatus |
| 设置静音状态 | NETDEV\_SetMuteStatus |
| 开启声音 | NETDEV\_OpenSound |
| 调节扬声器音量 | NETDEV\_SoundVolumeControl |
| 获取扬声器音量 | NETDEV\_GetSoundVolume |
| 关闭声音 | NETDEV\_CloseSound |
| 开启输入语音数据服务 | NETDEV\_StartInputVoiceSrv |
| 关闭输入语音数据服务 | NETDEV\_StopInputVoiceSrv |
| 输入语音数据 | NETDEV\_InputVoiceData |
| 设置解析后语音数据回调函数 | NETDEV\_SetParseVoiceDataCB |
| 获取语音广播组通道的状态 | NETDEV\_GetVoiceBroadcastChlStatus |
| 新建语音广播组 | NETDEV\_CreateVoiceBroadcastGroup |
| 修改语音广播组 | NETDEV\_ModifyVoiceBroadcastGroup |
| 开启语音广播 | NETDEV\_StartVoiceBroadcast |
| 关闭语音广播 | NETDEV\_StopVoiceBroadcast |
| 实况抓拍 | NETDEV\_CapturePicture |
| 动态产生一个关键帧 | NETDEV\_MakeKeyFrame |
| 本地录像 | NETDEV\_SaveRealData |
| 停止本地录像 | NETDEV\_StopSaveRealData |
| 根据文件类型.时间查找设备录像文件 | NETDEV\_FindFile |
| 逐个获取查找到的文件信息 | NETDEV\_FindNextFile |
| 关闭文件查找,释放资源 | NETDEV\_FindClose |
| 按时间回放录像文件 | NETDEV\_PlayBackByTime |
| 控制录像回放的状态 | NETDEV\_PlayBackControl |
| 停止回放业务 | NETDEV\_StopPlayBack |
| 按时间下载录像文件 | NETDEV\_GetFileByTime |
| 停止下载录像文件 | NETDEV\_StopGetFile |
| 打开本地文件 | NETDEV\_OpenMediaFile |
| 播放本地文件 | NETDEV\_StartPlayMediaFile |
| 停止播放本地文件业务 | NETDEV\_StopPlayMediaFile |
| 获取媒体文件时间总长度 | NETDEV\_GetMediaFileTime |
| 用户登录云端账户 | NETDEV\_LoginCloud |
| 云台控制操作(不启动实况预览) | NETDEV\_PTZControl\_Other |
| 获取云台预置位列表 | NETDEV\_GetPTZPresetList |
| 云台预置位操作(不需要启动预览) | NETDEV\_PTZPreset\_Other |
| 获取云台预置位巡航路径 | NETDEV\_PTZGetCruise |
| 云台预置位巡航操作(不需要启动预览) | NETDEV\_PTZCruise\_Other |
| 获取云台轨迹巡航路径 | NETDEV\_PTZGetTrackCruise\_V30 |
| 云台轨迹巡航操作(不需要启动预览) | NETDEV\_PTZTrackCruise\_V30 |
| 拉框放大.缩小(不需要启动预览) | NETDEV\_PTZSelZoomIn\_Other |
| 云台3D定位(不需要启动预览) | NETDEV\_PTZ3DPosition\_Other |
| 获取客流量统计信息列表 | NETDEV\_FindTrafficStatisticInfoList |
| 逐个查询获取到的客流量信息 | NETDEV\_FindNextTrafficStatisticInfo |
| 关闭查找，释放资源 | NETDEV\_FindCloseTrafficStatisticInfo |
| 获取客流量统计进度 | NETDEV\_GetTrafficStatisticProgress |
| 多通道开始客流量统计查询 | NETDEV\_StartMultiTrafficStatistic |
| 停止客流量查询 | NETDEV\_StopTrafficStatistic |
| 查看门禁授权组列表 | NETDEV\_FindACSPermissionGroupList |
| 获取下一条记录 | NETDEV\_FindNextACSPermissionGroupInfo |
| 关闭查询记录资源 | NETDEV\_FindCloseACSPermissionGroupList |
| 添加人员授权组信息 | NETDEV\_AddACSPersonPermissionGroup |
| 修改人员授权组信息 | NETDEV\_ModifyACSPersonPermissionGroup |
| 删除人员授权组信息 | NETDEV\_DeleteACSPersonPermissionGroup |
| 获取单个授权组信息 | NETDEV\_GetSinglePermGroupInfo |
| 访客签离 | NETDEV\_ACSPersonSignout |
| 获取指定人员授权信息 | NETDEV\_GetACSPersonPermission |
| 设置指定人员授权信息 | NETDEV\_SetACSPersonPermission |
| 查询出入记录 | NETDEV\_FindACSAttendanceLogList |
| 获取下一条出入记录 | NETDEV\_FindNextACSAttendanceLog |
| 关闭查询出入记录资源 | NETDEV\_FindCloseACSAttendanceLogList |
| 添加访客黑名单 | NETDEV\_AddACSPersonBlackList |
| 删除访客黑名单 | NETDEV\_DeleteACSPersonBlackList |
| 修改访客黑名单信息 | NETDEV\_ModifyACSPersonBlackList |
| 获取指定访客黑名单信息 | NETDEV\_GetACSPersonBlackList |
| 获取访客黑名单列表 | NETDEV\_FindACSPersonBlackList |
| 获取下一条记录 | NETDEV\_FindNextACSPersonBlackListInfo |
| 关闭查询记录资源 | NETDEV\_FindCloseACSPersonBlackList |
| 门禁通道控制 | NETDEV\_DoorCtrl |
| 门禁通道批量控制 | NETDEV\_DoorBatchCtrl |
| 门禁人员管理 | NETDEV\_ACSPersonCtrl |
| 批量添加人员信息 | NETDEV\_AddACSPersonList |
| 批量删除门禁人员信息 | NETDEV\_DeleteACSPersonList |
| 获取门禁人员信息列表 | NETDEV\_FindACSPersonList |
| 获取下一个门禁人员信息 | NETDEV\_FindNextACSPersonInfo |
| 关闭门禁人员信息列表资源 | NETDEV\_FindCloseACSPersonInfo |
| 获取指定人员所持门禁卡信息 | NETDEV\_GetACSPersonCardInfo |
| 绑定指定人员的门卡信息（即开卡） | NETDEV\_AddACSPersonCardInfo |
| 删除指定人员的门卡信息(即退卡) | NETDEV\_DeleteACSPersonCardInfo |
| 查询访客记录 | NETDEV\_FindACSVisitLogList |
| 获取下一条访客记录 | NETDEV\_FindNextACSVisitLog |
| 关闭查询访客记录资源 | NETDEV\_FindCloseACSVisitLog |
| 接收告警消息的回调函数 | NETDEV\_SetAlarmFGCallBack |
| 下发第三方支付订单 | NETDEV\_IssuePayOrder |
| 注册回调函数，接收停车场事件信息等 | NETDEV\_SetParkEventCallBack |
| 批量获取停车场信息 | NETDEV\_GetParkInfoList |
| 获取某停车场下所有出入口信息 | NETDEV\_GetParkEntranceInfoList |
| 获取停车场出入口所有车道信息 | NETDEV\_GetParkLaneInfoList |
| 创建人员库信息 | NETDEV\_CreatePersonLibInfo |
| 修改人员库信息 | NETDEV\_ModifyPersonLibInfo |
| 查询所有已创建的人员库信息 | NETDEV\_FindPersonLibList |
| 逐个获取查找到的 人脸库 信息 | NETDEV\_FindNextPersonLibInfo |
| 关闭查找 人脸库，释放资源 | NETDEV\_FindClosePersonLibList |
| 删除指定的人员库 | NETDEV\_DeletePersonLibInfo |
| 新增指定的人员信息 | NETDEV\_AddPersonInfo |
| 修改指定的人员信息 | NETDEV\_ModifyPersonInfo |
| 批量删除人员信息 | NETDEV\_DeletePersonInfoList |
| 查询所有人脸布控任务 | NETDEV\_FindPersonMonitorList |
| 逐个获取查找到的 布控任务 信息 | NETDEV\_FindNextPersonMonitorInfo |
| 关闭查找 布控任务，释放资源 | NETDEV\_FindClosePersonMonitorList |
| 新增单个人脸布控任务 | NETDEV\_AddPersonMonitorInfo |
| 删除单个人脸布控任务 | NETDEV\_DeletePersonMonitorInfo |
| 查询单个人脸布控任务配置信息 | NETDEV\_GetPersonMonitorRuleInfo |
| 设置单个人脸布控任务配置信息 | NETDEV\_SetPersonMonitorRuleInfo |
| 查询所有人员库的容量信息 | NETDEV\_GetPersonLibCapacity |
| 条件查询人员信息 | NETDEV\_FindPersonInfoList |
| 逐个获取查找到的 人员 信息 | NETDEV\_FindNextPersonInfo |
| 关闭查找 人员信息，释放资源 | NETDEV\_FindClosePersonInfoList |
| 注册人脸报警消息回调函数 | NETDEV\_SetPersonAlarmCallBack |
| 注册结构化报警消息回调函数 | NETDEV\_SetStructAlarmCallBack |
| 批量删除人脸布控任务 | NETDEV\_BatchDeletePersonMonitorInfo |
| 查询人脸识别记录 | NETDEV\_FindFaceRecordDetailList |
| 逐个获取查找到的 人脸识别记录信息 | NETDEV\_FindNextFaceRecordDetail |
| 关闭查找 人脸识别记录，释放资源 | NETDEV\_FindCloseFaceRecordDetail |
| 查询单个人脸识别记录的人脸图片信息 | NETDEV\_GetFaceRecordImageInfo |
| 获取系统图片信息 | NETDEV\_GetSystemPicture |
| 注册回调函数，接收车牌识别报警消息等 | NETDEV\_SetVehicleAlarmCallBack |
| 条件查询车辆成员详细信息 | NETDEV\_FindVehicleMemberDetailList |
| 逐个获取查找到的 车辆成员 信息 | NETDEV\_FindNextVehicleMemberDetail |
| 关闭查找 车辆成员，释放资源 | NETDEV\_FindCloseVehicleMemberDetail |
| 批量添加车辆成员信息 | NETDEV\_AddVehicleMemberList |
| 批量删除车辆成员信息 | NETDEV\_DelVehicleMemberList |
| 删除指定车辆成员信息 | NETDEV\_DelVehicleInfo |
| 获取单个车辆成员详细信息 | NETDEV\_GetVehicleMemberInfo |
| 修改指定车辆库中车辆信息 | NETDEV\_ModifyVehicleMemberInfo |
| 查询车辆识别的所有布控任务 | NETDEV\_FindVehicleMonitorList |
| 逐个获取查找到的车辆布控任务信息 | NETDEV\_FindNextVehicleMonitorInfo |
| 关闭查找车辆布控任务，释放资源 | NETDEV\_FindCloseVehicleMonitorList |
| 新增单个车辆布控任务 | NETDEV\_AddVehicleMonitorInfo |
| 批量删除车辆布控任务 | NETDEV\_DeleteVehicleMonitorInfo |
| 查询单个车辆布控任务配置信息 | NETDEV\_GetVehicleMonitorInfo |
| 设置单个车辆布控任务配置信息 | NETDEV\_SetVehicleMonitorInfo |
| 条件查询车辆识别记录的详细信息 | NETDEV\_FindVehicleRecordInfoList |
| 逐个获取查找到的车辆识别记录信息 | NETDEV\_FindNextVehicleRecordInfo |
| 关闭查找车辆识别记录，释放资源 | NETDEV\_FindCloseVehicleRecordList |
| 查询单个车辆识别记录的车辆图片信息 | NETDEV\_GetVehicleRecordImageInfo |
| 向指定的车辆库中批量划归车辆成员 | NETDEV\_AddVehicleLibMember |
| 批量取消指定的车辆库中车辆成员划归 | NETDEV\_DeleteVehicleLibMember |
| 查询全部车辆库信息列表 | NETDEV\_FindVehicleLibList |
| 逐个获取查找到的 车辆库 信息 | NETDEV\_FindNextVehicleLibInfo |
| 关闭查找 车辆库，释放资源 | NETDEV\_FindCloseVehicleLibList |
| 新增单个车辆库信息 | NETDEV\_AddVehicleLibInfo |
| 修改指定的车辆库信息 | NETDEV\_ModifyVehicleLibInfo |
| 删除指定的车辆库信息 | NETDEV\_DeleteVehicleLibInfo |
| 添加时间模板 | NETDEV\_AddTimeTemplate |
| 批量删除指定的时间模板 | NETDEV\_BatchDeleteTimeTemplate |
| 修改指定事件模板信息 | NETDEV\_ModifyTimeTemplate |
| 获取指定时间模板信息 | NETDEV\_GetTimeTemplate |
| 批量获取指定模板类型的时间模板信息 | NETDEV\_FindTimeTemplateByTypeList |
| 逐个获取按类型找到的时间模板信息 | NETDEV\_FindNextTimeTemplateByTypeInfo |
| 关闭查找 按类型找到的时间模板链表，释放资源 | NETDEV\_FindCloseTimeTemplateByTypeList |
| 订阅智能事件 | NETDEV\_SubscribeSmart |
| 取消订阅智能事件 | NETDEV\_UnsubscribeSmart |
| LAPI告警订阅 | NETDEV\_SubscibeLapiAlarm |
| 取消LAPI告警订阅 | NETDEV\_UnSubLapiAlarm |

# 帮助

## 编程环境配置

### Windows下SDK开发编译环境

* **SDK开发包目录说明**

SDK开发包包含如下文件夹：

include：该文件夹下包含SDK开发所需的头文件；

lib：该文件夹下包含SDK开发所需的lib库；

dll：该文件夹下包含SDK开发所需的动态库;

doc：该文件夹下包含SDK的使用手册;

demo：该文件夹下包含使用MFC、C#开发的demo源码;

bin：该文件夹下包含demo的可执行程序。

* **设置集成开发环境的编译项**

以VS2008集成开发环境为例进行说明，如SDK开发包路径：C:\NetDEVSDK；

（1）选择“项目”→“属性”菜单项，打开“属性页”窗口，选择其中“配置属性”→“C/C++”→“常规”， 在“附加包含目录”中输入“ C:\NetDEVSDK\include”；

（2）选择“配置属性”→“链接器”→“常规”，在“附加库目录”中输入“C:\NetDEVSDK\lib”；

（3）选择“配置属性”→“链接器”→“输入”，在附加依赖项中输入必需的lib库。

* **特别说明：**

SDK包含的所有动态库文件、配置文件和应用程序请放置在同一目录下使用。如将dll下的所有文件放置到执行程序所在目录下，如：NetDemo.exe所在目录。

SDK的日志输出名为“netdevsdk.log”。

* **依赖的库和头文件**

|  |  |  |
| --- | --- | --- |
| 头文件 | 库文件(Windows) | 备注 |
| NetDEVSDK.h | NetDEVSDK.lib |  |

### Linux下SDK开发编译环境

* **SDK开发包目录说明**

include：该文件夹下包含SDK开发所需的头文件；

lib：该文件夹下包含SDK开发所需的Linux库；

doc：该文件夹下包含SDK的使用手册;

demo：该文件夹下包含demo源码；

* **配置环境变量**

（1）使用cd命令进入lib目录下，复制lib目录的路径；

（2）将lib目录路径添加到vi /etc/ld.so.conf文件中并保存退出文件；

（3）执行ldconfig命令使环境变量生效；

### Android开发编译环境

* **SDK开发包目录说明**

include：该文件夹下包含SDK开发所需的头文件；

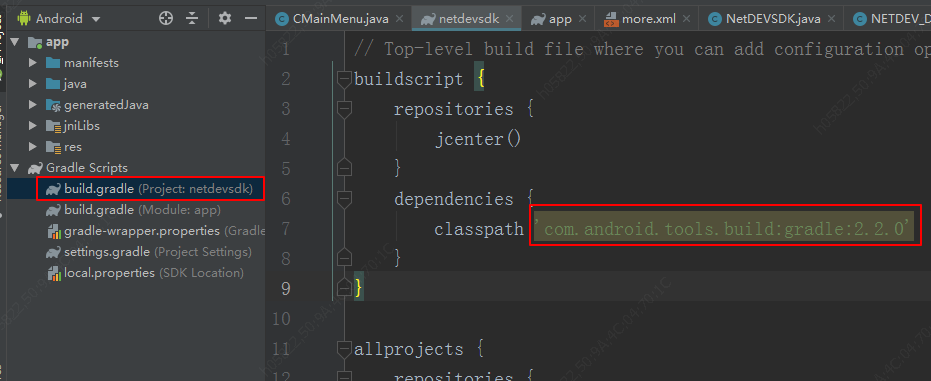
lib：该文件夹下包含SDK开发所需的Android库（32位及64位）；

doc：该文件夹下包含SDK的使用手册;

demo：该文件夹下包含Android Demo源码；

* **Demo配置**

（1）配置Build.gradle(Project:netdevsdk)，将gradle版本修改为用户的版本号。



（2）配置Build.gradle(Module:app)，将SDK版本修改为用户的版本号。



（3）local.properties配置，将sdk路径修改为用户本机路径。

### IOS开发编译环境

* **SDK开发包目录说明**

include：该文件夹下包含SDK开发所需的头文件；

lib：该文件夹下包含SDK开发所需的Linux库；

doc：该文件夹下包含SDK的使用手册;

demo：该文件夹下包含demo源码；

* **Xcode配置**

1. 添加SDK 库及其他依赖库。将解压后的压缩包中的lib 文件拷贝到项目路径下，在Build Phases 的Link Binary With Libraries 中添加SDK 静态库文件。同时添加iPhone SDK 依赖文件。SDK 静态库文件：lib 目录下所有.a 文件。其它文件：libiconv.tbd、libstdc++.tbd、libz.tbd、libc++.tbd。
2. 配置build Settings。设置Enable Bitcode值为NO，设置Enable Testability值为NO；

## 注意事项

* 优先使用带”\_V30”后缀的接口与结构体；

V30对函数和结构体做了优化，产品兼容性更好，推荐使用最新版本的函数或结构体；

* 更新SDK版本，所有库文件、头文件都需要更新；
* 除部分特殊说明的接口外，所有接口都必须在SDK初始化之后调用；
* 抓图及录像下载时，传入的文件路径所在目录必须存在，否则会调用接口失败；
* 控件SDK使用前，必须先以管理员权限安装控件SDK；

## 常见问题解答

1. SDK登录端口是多少？

* SDK登录端口与设备web访问端口一致，如设备web访问默认端口80，SDK登录默认端口也为80。

1. 如何判断回放/下载结束？

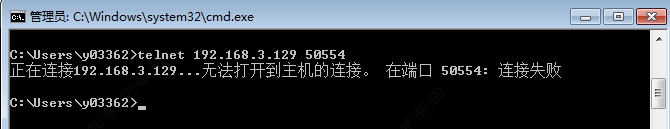
* 上层调用SDK接口回放/下载时，会下发对应的开始结束时间。
* 调用SDK接口获取当前回放/下载时间戳。
* 判断下发结束时间是否与当前时间戳一样，如果一样或者大于结束时间则下载结束。

1. SDK是否支持接入第三方厂家设备？

* SDK不支持接入第三方厂家设备。

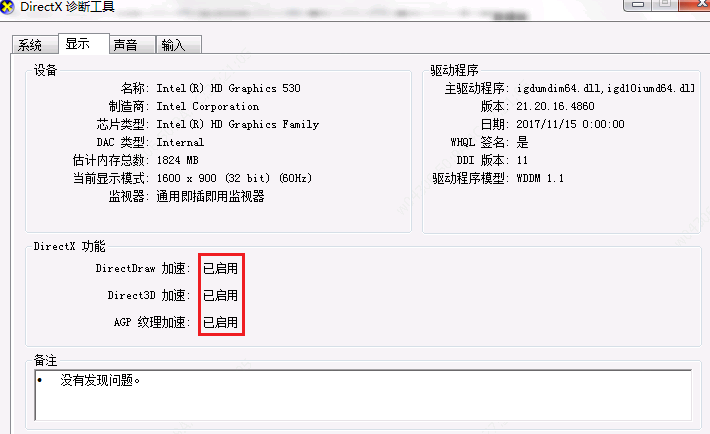
1. 局域网下，通过SDK启流成功，映射到公网后web启流成功且通过SDK登录成功，为何启流失败？

* Web与SDK启流方式不一样。
* SDK登录使用HTTP端口，启流使用RTSP端口。
* SDK启流失败，主要由于RTSP端口不通导致。可以使用命令分别测试HTTP、RTSP端口，确保端口映射正常，如下图端口不通：telnet 192.168.3.129 50554



1. WIN系统下通过SDK启流后，为何实况画面花屏或者绿屏？

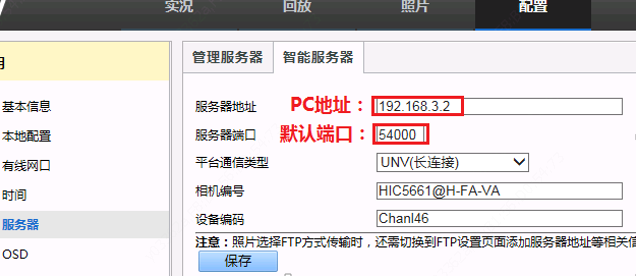
* 主要由于显卡驱动过老或者DirectX功能未开启。可以通过DirectX诊断工具显示信息进一步判断（Win+R运行dxdiag），正常情况如下图：



* 常用解决方案，升级显卡驱动。

1. 对接相机人脸抓拍功能，通过注册NETDEV\_SetFaceSnapshotCallBack回调，为何无法收到人脸图片数据？

* 在IE中开启人脸抓拍功能。
* 关闭电脑防火墙；
* 配置智能服务器PC地址与端口（SDK默认监听54000），如下图配置。



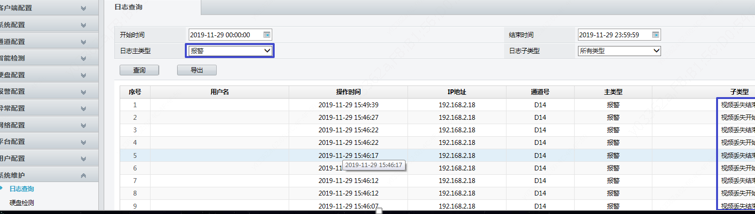
1. SDK非预览抓拍接口，为何抓取图片失败？

* 需要确保相机支持，在IPC的WEB界面查看，抓拍使能是否打开，如下图：



1. SDK对接告警功能，为何无法接收到告警信息？

* 关闭电脑防火墙。
* 检查web日志中是否有告警记录，确保真正触发。如下图NVR中告警记录查询：



1. Linux下集成SDK库，为何编译始终报链接错误？

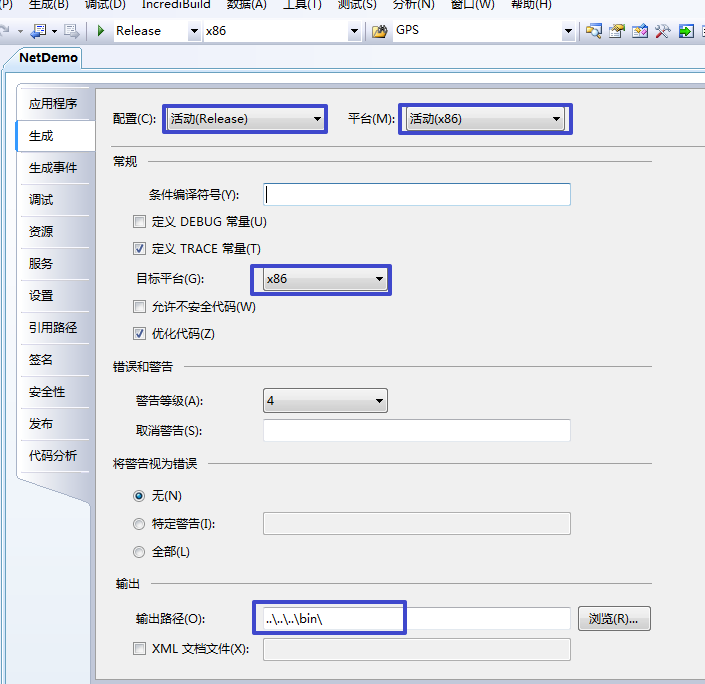
* 可以在/etc/ld.so.conf文件中追加添加SDK库所在绝对路径。
* 执行ldconfig命令。

1. 为何运行MFC Demo提示应用程序无法启动，因为应用程序的并行配置不正确…？

* 安装vc2008运行库即可，因MFC运行需依赖该系统环境。百度下载对应vcredist安装即可。

1. 为何C# Demo源码编译运行，无法加载到SDK库？

* 程序输出路径中是否包含所有的SDK dll动态库。
* 检测编译工程配置是否正确，与下载版本位数是否一致，如下图32位版本配置：



1. 通过SDK获取的数据中，为何中文显示乱码？

* SDK接口中字符串数据均是UTF8格式，中文需要转换才能正常显示。

1. 使用NETDEV\_SetDevConfig ()配置参数要注意什么？

* 因为NETDEV\_SetDevConfig ()需要输入完整赋值的结构体，否则容易导致设置出错。所以为了避免这个问题，一般地，在调用NETDEV\_SetDevConfig ()设置之前先调用NETDEV\_GetDevConfig ()给需要修改的结构体赋一个初始值，在此基础上修改对应参数，最后再调用NETDEV\_SetDevConfig ()。

1. 语音数据转发是什么？主要用于什么情况下？

* 语音数据转发，是语音对讲的网络部分功能，即接收和发送声音数据部分，可以将设备的声音回调处理，同时也可以调用NETDEV\_InputVoiceData ()发送音频数据到设备。
* 音频数据的来源比语音对讲更加自如，可以来自自行采集、声音文件等。

1. 实况、语音对讲的回调是否可以设置为NULL？

* 可以，这样实况、语音对讲依然正常，但是用户无法自行处理数据。

1. 语音广播的功能？

* 语音广播是指向多台设备发送需要广播的音频数据，但是对设备发过来的声音不作处理；一台设备如果只有一个语音输入口，同时只能在进行语音广播、语音对讲、语音转发中的一种。

1. 实时预览过程必须调用的接口及顺序？

* 正确调用顺序如下：NETDEV\_Init——> NETDEV\_Login\_V30——> NETDEV\_RealPlay——> NETDEV\_StopRealPlay——> NETDEV\_Logout ——> NETDEV\_Cleanup。