设备网络 SDK 用户开发手册

Issue V03

目 录

1 概述

网络 SDK 是软件开发者在开发网络硬盘录像机、网络视频服务器、网络摄像机、网络球机、智能设备等产品监控联网应用时的开发套件。本文档详细描述了开发包中各个函数实现的功能、接口及其函数之间的调用关系和示例实现。

1.1 功能概述

本开发套件主要包括业务操作和设备管理两大部分：

业务操作

状态侦听、实时监视、实时预览、字符叠加、音频控制、录像回放和下载、数据保存、云台控制、语音对讲、透明串口、码流统计、智能分析等功能。

设备管理

远程升级、远程重启、设备参数配置（系统通用配置、报警布/撤防设置、录像配置、串口配置、图像配置、日志管理、用户管理、设备校时、动态检测配置、网络配置）等功能。

1.2 SDK 组件

1.2.1 头文件

common.h 包含 SDK 使用到的宏定义和结构体定义

IPCNetSDK_Interface.h 包含 SDK 所有的 API 定义

1.2.2 静态链接库

IPCNetSDK.lib

1.2.3 动态库

IPCNetSDK.dll

IPCPlayer.dll

1.2.4 文档

《设备网络 SDK 开发手册》

2 宏、枚举、结构体、接口定义

2.1 宏定义

宏定义	宏定义值	含义
MANUFACTURER	""	制造商英文名
MAX_DEVICE_NAME_LEN	127	设备名长度
MAX_SERIAL_NUM	127	设备序列号长度
MAX_USERNAME_LEN	63	用户名长度
MAX_LOCATION_LEN	127	路径长度
MAX_DOMAIN_NAME	127	域名长度
MAX_PWD_LEN	15	用户密码长度
MAX_IPC_CHN_NUM	1	网络摄像机通道数
MAX_DAYS	7	每周的天数
MAX_TIMESEGMENTS	6	每天时间段数
IPADRRESS_LEN	16	IPv4 地址长度
IPV6ADDRESS_LEN	128	IPv6 地址长度
MAX_IPFILTER_NUM	8	IP 地址过滤条目数
MAC_LEN	23	Mac 地址长度
MAX_USER_NUM	16	用户数量
MAX_OSDTXT_LEN	127	自定义 OSD 字符长度
MAX_MD_AREAS	4	移动侦测区域数
MAX_PRIVACYMASK_AREAS	4	隐私遮蔽区域数
MAX_IOIN_NUM	64	最大报警输入数
MAX_IOOUT_NUM	64	最大报警输出数
MAX_NET_PROTO_NUM	32	最大支持协议数

MAX_SSID_LEN	127	WIFI 的 SSID 号长度
MAX_WIFIKEY_LEN	127	最大密钥长度
MAX_EMAILADDR_LEN	63	邮箱地址长度
MAX_MAIL_RECEIVERS	4	接受邮箱数
MAX_EMAILSRV_NUM	2	邮件发送服务器数
MAX_ALARMOUT	4	最大报警输出端口数
MAX_CHANNUM	16	最大报警通道或录像通道数
MAX_DISKNUM	16	最大硬盘数
MAX_MAX_PTZPRESET_NUM	8	最大预置位数

【注意】 字符长度不包含字符结束符'\0'。

2.2 枚举定义

2.2.1 权限类型

```
typedef enum enumPrivilegeType

{

   TYPE_OPERATE      = 0,

   TYPE_CONFIGURE    = 1,

} ePrivilegeType;
```

名称	说明
TYPE_OPERATE	操作权限类型
TYPE_CONFIGURE	配置权限类型

2.2.2 用户等级

```
typedef enum enumUserLevel

{

   USER_LEVEL_ADMIN      = 0,

   USER_LEVEL_OPERATOR   = 1,

   USER_LEVEL_USER       = 2

} eUserLevel;
```

名称	说明
USER_LEVEL_ADMIN	管理员，拥有所有权限
USER_LEVEL_OPERATOR	操作员，可分配除用户管理以外的权限
USER_LEVEL_USER	观看者，没有任何操作和配置权限，只能接入视频预览

2.2.3 网络协议类型

```
typedef enum enumVSNetProtocal

{

   PROTO_I8    = 0x1,

   PROTO_VSIP  = 0x2,

   PROTO_ONVIF = 0x4,

   PROTO_28181 = 0x8

} eVSNetProtocal;
```

名称	说明
PROTO_I8	AE I8 协议

PROTO_VSIP	CMS VSIP 协议
PROTO_ONVIF	ONVIF 协议
PROTO_28181	国标 28181 协议

【注意】 协议可多选

2.2.4 IP 地址过滤类型

```
typedef enum enumIPFilterType
{
   IPFILTER_ALLOW       = 0,
   IPFILTER_DENY        = 1,
   IPFILTER_NONE        = 2
} eIPFilterType;
```

名称	说明
IPFILTER_ALLOW	白名单
IPFILTER_DENY	黑名单

2.2.5 OSD 位置类型

```
typedef enum enumOSDPos
{
   OSD_TOP_LEFT = 0,
   OSD_BOTTOM_LEFT,
   OSD_TOP_RIGHT,
   OSD_BOTTOM_RIGHT,
```

```
    OSD_POS_XY

} eOSDPos;
```

名称	说明
OSD_TOP_LEFT	左上角
OSD_BOTTOM_LEFT	左下角
OSD_TOP_RIGHT	右上角
OSD_BOTTOM_RIGHT	右下角
OSD_POS_XY	通过坐标指定位置

2.2.6 码率控制类型

```
typedef enum enumRateControlType

{

    RC_VBR = 0,

    RC_CBR,

    RC_CVBR,

    RC_OFF

} eRateControlType;
```

名称	说明
RC_VBR	变码率，不建议使用
RC_CBR	定码率
RC_CVBR	限制变码率

RC_OFF	无码率控制

2.2.7 码流传输类型

```
typedef enum enumTransportType

{

   RTPoverUDP = 0,

   RTPoverTCP = 1,

   TSoverTCP

} eTransportType;
```

名称	说明
RTPoverUDP	RTP 封包格式通过 UDP 传输
RTPoverTCP	RTP 封包格式通过 RTSP/TCP 传输
TSoverTCP	私有封包格式通过 TCP 传输，暂只支持本模式

2.2.8 OSD 日期显示格式

```
typedef enum enumDateFormat

{

   FMT_YMD = 0,

   FMT_MDY,

   FMT_DMY

} eDateFormat;
```

名称	说明

FMT_YMD	年月日
FMT_MDY	月日年
FMT_DMY	日月年

2.2.9 OSD 时间显示格式

```
typedef enum enumTimeFormat

{

   FMT_HMS = 0,   ///< hour minute second

   FMT_SMH       ///< second minute hour

} eTimeFormat;
```

名称	说明
FMT_HMS	时分秒
FMT_SMH	秒分时

2.2.10 OSD 小时格式

```
typedef enum enumHourFormat

{

   FMT_24HR = 0,

   FMT_12HR

} eHourFormat;
```

名称	说明
FMT_24HR	24 小时制

FMT_12HR	12 小时制

2.2.11 日夜模式

```
typedef enum enumDayNightFilterType

{

   DAYNIGHTFILTER_AUTO   = 0,

   DAYNIGHTFILTER_DAY    = 1,

   DAYNIGHTFILTER_NIGHT  = 2,

   DAYNIGHTFILTER_CUSTOM

} eDayNightFilterType;
```

名称	说明
DAYNIGHTFILTER_AUTO	自动切换日夜模式
DAYNIGHTFILTER_DAY	白天模式
DAYNIGHTFILTER_NIGHT	夜晚模式
DAYNIGHTFILTER_CUSTOM	用户自定义模式

2.2.12 IP 地址版本

```
typedef enum enumIPVersion

{

   IPV4    = 0x1,

   IPV6    = 0x2,

   IPVDUAL = 0x3
```

```
} eIPVersion;
```

名称	说明
IPV4	IP 地址结构体仅包含 v4 版本
IPV6	IP 地址结构体仅包含 v6 版本
IPVDUAL	IP 地址结构体同时包含 v4 和 v6 版本

2.2.13 日志类型

```
typedef enum enumLogType

{

   LOGTYPE_SYSTEM = 0,

   LOGTYPE_ACCESS = 1,

   LOGTYPE_ALARM  = 2,

    LOGTYPE_COUNT

} eLogType;
```

名称	说明
LOGTYPE_SYSTEM	系统日志，记录设备运行状态信息
LOGTYPE_ACCESS	操作日志，记录用户操作
LOGTYPE_ALARM	报警日志，记录报警信息

2.2.14 系统日志子类型

```
typedef enum enumSysLogSubType

{
```

```
    SYS_START           = 0,
    SYS_NTP_UPDATETIME      = 1,
    SYS_STOP,
    SYS_REBOOT_ABNORMAL,
    SYS_FACTORY_RESET,
    SYS_UPGRADE,
    SYS_SDINSERD,
    SYS_FORMATSD,
    SYS_SDREMOVED,
    SYS_START_SNAP,
    SYS_STOP_SNAP,
    SYS_START_REC,
    SYS_STOP_REC,
    SYS_REMOVE_OLDEST_REC,
    SYS_REMOVE_OLDEST_PIC,
    SYS_START_UPLOAD,
    SYS_STOP_UPLOAD,
    SYS_IPCHANGED,
} eSysLogSubType;
```

名称	说明
SYS_START	系统启动

SYS_NTP_UPDATETIME	
SYS_STOP	
SYS_REBOOT_ABNORMAL	
SYS_FACTORY_RESET	
SYS_UPGRADE	
SYS_SDINSERD	
SYS_FORMATSD	
SYS_SDREMOVED	
SYS_START_SNAP	
SYS_STOP_REC	
SYS_REMOVE_OLDEST_REC	
SYS_REMOVE_OLDEST_PIC	
SYS_START_UPLOAD	
SYS_STOP_UPLOAD	
SYS_IPCHANGED	

2.2.15 操作日志子类型

```
typedef enum enumAccessLogSubType

{

} eAccessLogSubType;
```

名称	说明

2.2.16 DDNS 服务器类型

```
typedef enum enumDDNSServer

{

   DDNS_3322 = 0,

DDNS_ORAY,

 DDNS_COUNT

} eDDNSServer;
```

名称	说明
DDNS_3322	3322 服务器
DDNS_ORAY	花生壳服务器

2.2.17 OSD 字体大小

```
typedef enum enumFontSize

{

   FS_32x32 = 0,

   FS_24x24,

   FS_16x16,

   FS_12x12,

   FS_10x10,

   FS_8x8,

   FS_COUNT
```

```
} eFontSize;
```

名称	说明
FS_32x32	
FS_24x24	
FS_16x16	
FS_12x12	
FS_10x10	
FS_8x8	

2.2.18 视频编码格式

```
typedef enum enumVideoCodecType

{

   VIDCODEC_INVALID = -1,

   VIDCODEC_H264    = 0,

   VIDCODEC_MPEG4   = 1,

   VIDCODEC_MJPEG   = 2,

   VIDCODEC_H265    =3,

   VIDCODEC_COUNT

} eVideoCodecType;
```

名称	说明
VIDCODEC_H264	H.264 视频编码

VIDCODEC_MPEG4	MPEG4 视频编码
VIDCODEC_MJPEG	MJPEG 视频编码
VIDCODEC_H265	H.265 视频编码

2.2.19 音频编码格式

```
typedef enum enumAudioCodecType

{

   AUDCODEC_INVALID   = -1,

   AUDCODEC_G711_ULAW,

   AUDCODEC_G711_ALAW,

   AUDCODEC_COUNT

} eAudioCodecType;
```

名称	说明
AUDCODEC_G711_ULAW	g.711 ulaw
AUDCODEC_G711_ALAW	g.711 alaw

2.2.20 视频帧类型

```
typedef enum enumVideoFrameType

{

   FRAME_INTRA        = 0,

   FRAME_INTRA_ENCRYPT = 1,

   FRAME_INTER        = 2,
```

```
} eVideoFrameType;
```

名称	说明
FRAME_INTRA	无加密 I 帧
FRAME_INTRA_ENCRYPT	加密 I 帧，只针对特定加密机型有效
FRAME_INTER	非关键帧

2.2.21 自动曝光自动白平衡模式

```
typedef enum enumAEWBMode

{

   AEWB_MODE_OFF = 0,

   AEWB_MODE_AE,

   AEWB_MODE_AWB,

   AEWB_MODE_AEWB

} eAEWBMode;
```

名称	说明
AEWB_MODE_OFF	同时关闭自动曝光和自动白平衡
AEWB_MODE_AE	仅打开自动曝光
AEWB_MODE_AWB	仅打开自动白平衡
AEWB_MODE_AEWB	同时打开自动曝光和自动白平衡

2.2.22 白平衡模式

```
typedef enum enumWhiteBalanceMode
```

```
{
   WB_AUTO   = 0,
   WB_OUTDOOR,
   WB_INDOOR,
   WB_MANUAL
} eWhiteBalanceMode;
```

名称	说明
WB_AUTO	自动白平衡
WB_OUTDOOR	室内
WB_INDOOR	室外
WB_MANUAL	手动设置白平衡，不建议用户自设置白平衡

2.2.23 视频采集模式

```
typedef enum enumVideoCaptureMode
{
   CAPTURE_MODE_720P_VGA = 0,
   CAPTURE_MODE_720P_QVGA,
   CAPTURE_MODE_720P_720P,
   CAPTURE_MODE_1080P_D1,
   CAPTURE_MODE_1080P_QVGA,
   CAPTURE_MODE_2MEGA,
   CAPTURE_MODE_5MEGA,
```

```
    CAPTURE_MODE_960P_VGA,

    CAPTURE_MODE_960P_QVGA,

    CAPTURE_MODE_COUNT

} eVideoCaptureMode;
```

名称	说明
CAPTURE_MODE_720P_VGA	
CAPTURE_MODE_720P_QVGA	
CAPTURE_MODE_720P_720P	
CAPTURE_MODE_1080P_D1	

【注意】 每种机型支持的视频采集模式不同。可通过获取能力集接口获取视频采集能力集。

2.2.24 测光模式

```
typedef enum enumMETERMODE

{

    METER_TOP = 0,

    METER_BOTTOM,

    METER_LEFT,

    METER_RIGHT,

    METER_MATRIX,

    METER_CENTER

} eMETERMODE;
```

名称	说明
METER_TOP	上侧测光
METER_BOTTOM	下侧测光
METER_LEFT	左侧测光
METER_RIGHT	右侧测光
METER_MATRIX	全局测光
METER_CENTER	中心测光

2.2.25 IRCUT 模式

```
typedef enum enumIRCUTMode

{

   IRCUT_ON          = 0,

   IRCUT_OFF         = 1,

   IRCUTE_AUTO       = 2,

} eIRCUTMode;
```

名称	说明
IRCUT_ON	打开 IRCUT
IRCUT_OFF	关闭 IRCUT
IRCUTE_AUTO	IRCUT 自动切换

2.2.26 移动侦测敏感度

```
typedef enum enumMDSensitivity

{
```

```
    MD_LOW = 0,

    MD_MEDIUM,

    MD_HIGH

} eMDSensitivity;
```

名称	说明
MD_LOW	
MD_MEDIUM	
MD_HIGH	

2.2.27 时间同步模式

```
typedef enum enumTimeSetMode

{

    TIMESET_MANUAL = 0,

    TIMESET_NTP    = 1

} eTimeSetMode;
```

名称	说明
TIMESET_MANUAL	手动设置
TIMESET_NTP	同步 NTP 时间（需要设置 NTP 服务器）

2.2.28 报警处理方式

```
typedef enum enumAlarmOutType

{

    ALARM_OUT_IO    = 0x1,
```

```
    ALARM_OUT_LED   = 0x2,

    ALARM_OUT_SNAP  = 0x4,

    ALARM_OUT_REC   = 0x8,

    ALARM_OUT_FTP   = 0x10,

    ALARM_OUT_PTZ   = 0x20,

ALARM_OUT_TCP   = 0x40,

ALARM_OUT_EMAIL = 0x80

} eAlarmOutType;
```

名称	说明
ALARM_OUT_IO	IO 输出
ALARM_OUT_LED	报警 LED 灯输出
ALARM_OUT_SNAP	报警抓拍
ALARM_OUT_REC	报警录像
ALARM_OUT_FTP	报警 FTP 上传录像
ALARM_OUT_PTZ	PTZ 联动，仅配备云台设备的机型支持
ALARM_OUT_TCP	报警上传 TCP 报警服务器
ALARM_OUT_EMAIL	发送报警邮件

2.2.29 报警上报使能方式

```
eEvtEnbType typedef enum enumEventEnableType

{

    EET_UNUSED  = 0,
```

```
    EET_ALWALYS = 1,

    EET_SCHETAB = 2,

    EET_DISABLE = 3

} eEvtEnbType;
```

名称	说明
EET_UNUSED	预留、未使用
EET_ALWALYS	总是上报
EET_SCHETAB	定时上报
EET_DISABLE	不上报

2.2.30 用户注册登录回调函数上报信息类型

```
typedef enum enumInfoType

{

    INFOTYPE_ALARM,

    INFOTYEP_EXCEPTION

}eInfoType;
```

名称	说明
INFOTYPE_ALARM	上报告警，对应的回调信息结构为：ALARM_INFO
INFOTYEP_EXCEPTION	上报异常，对应的回调信息结构为：EXCEPTION_INFO

2.2.31 获取配置方式

```
typedef enum enumGetPara
```

```
{
   GET_PARA_NONE      = 0x0,
   GET_PARA_MEM       = 0x1,  /// Set Parameter to memory
   GET_PARA_INI       = 0x2   /// Set Parameter to ini file
} eGetPara;
```

名称	说明
GET_PARA_NONE	默认值。默认从 IPC 内存获取配置
GET_PARA_MEM	从 IPC 内存获取配置
GET_PARA_INI	从 IPC 硬存储保存的配置获取配置

2.2.32 设置配置方式

```
typedef enum enumSetPara
{
   SET_PARA_NONE      = 0x0,
   SET_PARA_MEM       = 0x1,  /// Set Parameter to memory
   SET_PARA_INI       = 0x2   /// Set Parameter to ini file
} eSetPara;
```

名称	说明
SET_PARA_NONE	默认值。默认保存配置到硬存储，并设为当前配置
SET_PARA_MEM	设配置为当前配置，但不保存到硬存储
SET_PARA_INI	硬存储配置

2.2.33 用户操作类型

```
typedef enum enumOperateType

{

   OPERATE_ADD_USER    = 0x0,

   OPERATE_MODIFY_USER = 0x1,

   OPERATE_DELETE_USER = 0x2

}eOperateType;
```

名称	说明
OPERATE_ADD_USER	添加用户
OPERATE_MODIFY_USER	修改用户
OPERATE_DELETE_USER	删除用户

2.2.34 设备扫描状态标志

```
typedef enum enumSearchCallbackFlag

{

   CallBackFlag_Result,

   CallBackFlag_End

}eSearchCallbackFlag;
```

名称	说明
CallBackFlag_Result	扫描到一个 IPC
CallBackFlag_End	设备扫描结束

2.3 结构定义

2.3.1 IP 地址

```
typedef struct tagIPAddr

{

    char      ipV4[IPADRRESS_LEN];

    char      ipV6[IPV6ADDRESS_LEN];

    int       ipver;

} IPAddr_t;
```

成员	描述	备注
ipV4	ipv4 地址	
ipV6	ipv6 地址	
ipver	IP 地址版本	见 eIPVersion

【注意】

2.3.2 时间信息

具体到 xx 年 xx 月 xx 日 xx 时 xx 分 xx 秒

```
typedef struct tagTimeInfo

{

   short int Year;

   short int Month;

   short int Day;

   short int Hour;

   short int Minute;
```

```
    short int Second;

} TimeInfo_t;
```

成员	描述
Year	年，起始年为 1970 年
Month	月，范围 1 到 12
Day	日，范围 1 到 31
Hour	小时，范围 0 到 23
Minute	分，范围 0 到 59
Second	秒，范围 0 到 59

【注意】

2.3.3 时间点

一天内的时间点

```
typedef struct tagTimePoint

{

    int tm_hour;

    int tm_min;

    int tm_sec;

} TimePoint_t;
```

成员	描述
tm_hour	小时，范围 0 到 23
tm_min	分，范围 0 到 59

tm_sec	秒，范围 0 到 59

2.3.4 时间段

```
typedef struct
{
    int         bEnable;
    TimePoint_t start;
    TimePoint_t end;
} TimeSection_t;
```

成员	描述	备注
bEnable	使能开关	
start	起始时间	
end	结束时间	

【注意】有效的时间段：起始时间必须小于结束时间

2.3.5 时间表

```
typedef struct
{
    int         bAllDay[MAX_DAYS];
    TimeSection_t section[MAX_DAYS][MAX_TIMESEGMENTS];
} TimeTable_t;
```

成员	描述	备注
bAllDay	全天时间有效	

section	一周时间段	

2.3.6 NTP 配置

```
typedef struct tagNtpCfg

{

   int    bEnable;

   int    timeZone;

   int    interval;

   int    ntpPort;

   char   ntpServiceName[MAX_HOST_NAME_LEN + 1];

} NtpCfg_t;
```

成员	描述	备注
bEnable	使能开关	
timeZone	时区	
interval	更新时间间隔	
ntpPort	ntp 服务端口	
ntpServiceName	ntp 域名	

2.3.7 矩形

```
typedef struct tagRectangle

{

   int startx;
```

```
    int starty;

    int width;

    int height;

} Rectangle_t;
```

成员	描述	备注
startx	左上角横坐标	
starty	左上角纵坐标	
width	宽	
height	高	

2.3.8 用户登录信息

```
typedef struct tagUserLoginInfo

{

    int         netType;

    IPAddr_t    userIPAddr;

    char        szUserName[MAX_USERNAME_LEN + 1];

    char        szPassword[MAX_PWD_LEN + 1];

    unsigned int port;

} UserLoginInfo_t;
```

成员	描述	备注
netType	登入网络类型	可忽略

userIPAddr	设备 IP 地址	
szUserName	用户登录名	多字节格式
szPassword	用户密码	多字节格式
port	设备服务端口号	

【注意】

2.3.9 设备基本信息

```
typedef struct tagDeviceInfo

{

    unsigned int softver;

    unsigned int hardver;

    unsigned int panelver;

    unsigned int devType;

    char        serialNum[MAX_SERIAL_NUM + 1];

    char        szDeviceName[MAX_DEVICE_NAME_LEN + 1];

    char        byMAC[MAC_LEN + 1];

    IPAddr_t     ip;

    unsigned int httpPort;

    unsigned int ftpPort;

    unsigned int rtspPort;

    unsigned int tcpPort;

    unsigned int updateTimes;
```

```
    char        alarminNum;

    char        alarmoutNum;

    char        maxChan;

    char        audioChan;

    char        resolution;

    char        wifi;

    char        sdslot;

    char        diskNum;

} DeviceInfo_t;
```

成员	描述	备注
softver	软件版本号	
hardver	硬件版本号	
panelver	面板版本号	暂未用
devType	设备类型	
serialNum	设备序列号	不可更改
szDeviceName	设备名	
byMAC	Mac 地址	不可更改
ip	IP 地址	
httpPort	web 服务端口号	
ftpPort	ftp 服务端口号	
rtspPort	rtsp 服务端口号	

tcpPort	tcp 服务端口号	
updateTimes	升级固件次数	
alarminNum	报警输入数	
alarmoutNum	报警输出数	
maxChan	最大通道数	IPC 为 1
audioChan	音频通道数	
resolution	最大分辨率	
wifi	是否支持 WIF	
sdslot	是否支持 SD 卡	
diskNum	硬盘个数	IPC 设备可忽略

2.3.10 用户信息

```
typedef struct tagUserInfo

{

    int          userLevel;

    char         szUserName[MAX_USERNAME_LEN + 1];

    char         szPassword[MAX_PWD_LEN + 1];

    unsigned int  cfgRight;

    unsigned int  opRight;

} UserInfo_t;
```

成员	描述	备注
userLevel	用户等级	

szUserName	用户名	不能为空
szPassword	用户密码	不能为空
cfgRight	配置权限	
opRight	操作权限	

2.3.11 用户信息列表

```
typedef struct tagUserList

{

   int       count;

   UserInfo_t User[MAX_USER_NUM];

} UserList_t;
```

成员	描述	备注
count	用户个数	
User	用户信息数组	

2.3.12 多播配置

```
typedef struct tagMulticastCfg

{

   IPAddr_t   struMulticastIpAddr;

   int       MulticastPort;

} MulticastCfg_t;
```

成员	描述	备注

struMulticastIpAddr	多播源地址	
MulticastPort	多播服务端口	

2.3.13 OSD 配置

```
typedef struct tagOSDCfg

{

   int    osdPosition;

   int    osdX;

   int    osdY;

   int    osdColor;

   int    osdFont;

   int    osdSize;

   int    osdSysInfoEnable;

   int    osdDateEnable;

   int    osdDateFormat;

   int    osdTimeEnable;

   int    osdTimeFormat;

   int    osdUsrTextEnable;

   char   osdUsrText[MAX_OSDTXT_LEN + 1];

   int    osdImgEnable;

   int    imgTransColor;

} OSDCfg_t;
```

成员	描述	备注
osdPosition	osd 显示位置	
osdX	osd 位置横坐标	
osdY	osd 位置纵坐标	
osdColor	osd 颜色	暂不支持
osdFont	字体类型	暂不支持
osdSize	字体大小	
osdSysInfoEnable	显示系统信息使能开关	内部调试使用
osdDateEnable	日期显示开关	
osdDateFormat	日期显示格式	
osdTimeEnable	时间显示开关	
osdTimeFormat	时间显示格式	
osdUsrTextEnable	用户自定义信息显示开关	
osdUsrText	用户自定义信息	
osdImgEnable	Logo 显示开关	
imgTransColor	Logo 透明色	暂不支持

2.3.14 日志查询条件

```
typedef struct tagLogQueryCond

{

   int        chn;

   TimeInfo_t  starttime;

   TimeInfo_t  endtime;
```

```
    short int   major;

    short int   minor;

} LogQueryCond_t;
```

成员	描述	备注
chn	通道号	IPC 设备可忽略
starttime	查询起始时间	
endtime	查询结束时间	
major	日志主类型	
minor	日志子类型	

2.3.15 日志条目

```
typedef struct tagLogEntry

{

    int         major;

    int         minor;

    TimeInfo_t  logtime;

    char        szUser[MAX_USERNAME_LEN + 1];

    IPAddr_t    remoteHostAddr;

    short int   alarmInPort;

    short int   alarmOutPort;

    int         channel;

    int         infoLen;

    char        szInfo[MAX_LOG_INFO_LEN + 1];
```

```
} LogEntry_t;
```

成员	描述	备注
major	日志主类型	
minor	日志子类型	
logtime	日志记录时间	
szUser	操作用户名	日志类型为操作日志时有效
remoteHostAddr	操作用户 IP 地址	日志类型为操作日志时有效
alarmInPort	报警输入端口	日志类型为报警日志时有效
alarmOutPort	报警输出端口	日志类型为报警日志时有效
channel	通道号	IPC 设备可忽略
infoLen	日志详细信息字符串长度	可选项
szInfo	日志详细信息字符串数组	可选项

2.3.16 报警上报规则

```
typedef struct
{
   unsigned int      uAlarmEnable;
   TimeTable_t       struSched;
   NET_ALARM_HANDLE  handle;
} NET_ALARM_PARAM;
```

成员	描述	备注
uAlarmEnable	是否上报或定时上报	见 eEvtEnbType 枚举定义

struSched	定时上报的时间表， uAlarmEnable 值为 EET_SCHETAB 时有效	见 TimeTable_t 结构定义
handle	报警上报方式	见 NET_ALARM_HANDLE 结构定义

2.3.17 摄像头配置

```
typedef struct tagCameraCfg

{

   int iaewbType;              ///< AEWB_MODE_OFF, AEWB_MODE_AE, AEWB_MODE_AWB,
AEWB_MODE_AEWB

   int iPowerLineFrequencyMode;   ///< 0:FLICKER_NTSC_MOD 1:FLICKER_PAL_MOD

   int iExposurePriority;        ///< 0:EXPOSUREPRIT_DEFAULT 1:EXPOSUREPRIT_MANUL

   int iExposureValue;           ///< [0..100]

   int iSaturation;             ///< [0..100]

   int iBrightness;             ///< [0..100]

   int iSharpness;              ///< [0..100]

   int iContrast;               ///< [0..100]

   int iDayNightFilterType;      ///< 0:DAYNIGHTFILTER_AUTO 1:DAYNIGHTFILTER_DAY
2:DAYNIGHTFILTER_NIGHT 3:DAYNIGHTFILTER_CUSTOM

   int iAutoIRISEnable;          ///< 0: AutoIRISDisable 1: AutoIRISEnable

   int iMeter;                  ///<  METER_MATRIX, METER_CENTER

   int iWhiteBalanceMode;        ///< 0:WB_AUTO, 1:WB_OUTDOOR, 2:WB_INDOOR,
3:WB_MANUAL

   int iRedGain;                ///< [0..100]
```

```
    int iBlueGain;              ///< [0..100]

    int iMirrorValue;           ///< [0..3] 0:flipH ON flipV ON, 1:flipH ON flipV OFF, 2:flipH OFF
flipV ON, 3:flipH OFF flipV OFF

    int iNoise;                 ///< [0..100]

    int iBacklight;             ///0 off,1 on

    int iLightInhibition;       ///0 off,1 on

    int iWideDynamic;           ///0 off,1 on

    int iShutter;

    int iLensDCLevel;                       ///< [0..100]

     int iCol2GreyDayLuma;                  ///< [0..100]

     int iCol2GreyNightLuma;                ///< [0..100]

     DaynightDuty_t    dnduty[MAX_DNDUTY_NUM];          //daynight mode duty table

} CameraCfg_t;
```

成员	描述	备注
iaewbType	自动曝光自动白平衡模式	AEWB_MODE_OFF, AEWB_MODE_AE, AEWB_MODE_AWB
iSaturation	饱和度	[0..100]
iBrightness	亮度	[0..100]
iSharpness	锐利度	[0..100]
iContrast	对比度	[0..100]
iWhiteBalanceMode	白平衡模式	0:WB_AUTO, 1:WB_OUTDOOR, 2:WB_INDOOR, 3:WB_MANUAL

iExposureValue	曝光值	暂时无效
iPowerLineFrequencyMode	扫描频率	0:FLICKER_NTSC_MOD 1:FLICKER_PAL_MOD
iMeter	测光模式	METER_MATRIX, METER_CENTER
iExposurePriority	曝光优先级	暂时无效
iAutoIRISEnable	自动光圈开关，0 为关闭，1 为打开	0: AutoIRISDisable 1: AutoIRISEnable
iRedGain	红增益,0-100	[0..100]
iBlueGain	蓝增益,0-100	[0..100]
iMirrorValue	镜像值	[0..3] 0:flipH ON flipV ON, 1:flipH ON flipV OFF, 2:flipH OFF flipV ON, 3:flipH OFF flipV OFF
iDayNightFilterType	日夜模式	0:DAYNIGHTFILTER_AUTO 1:DAYNIGHTFILTER_DAY 2:DAYNIGHTFILTER_NIGHT 3:DAYNIGHTFILTER_CUSTOM
iNoise	Noise,0-100	[0..100]
iBacklight	Backlight compensation,0-off,1-on	
iLightInhibition	Light inhibition,0-off,1-on	
iWideDynamic	Wide dynamic,0-off,1-on	
iShutter	Shutter,0-1/5,1-1/10,2-1/15,3-1/20,4-1/25(50hz)1/30(60hz),5-1/50(50hz)1/60(60hz),6-1/100(50hz)1/120(60hz),7-1/125,8-1/200,9-1/250,10-1/500,11-1/1000	
iLensDCLevel	镜头矫正	*[0..100]*
iCol2GreyDayLuma	内同步转彩色调节值	*[0..100]*

iCol2GreyNightLuma	内同步转黑白调节值	*[0..100]*
dnduty[MAX_DNDUTY_NUM]	定时切换时间表	参考 *DaynightDuty_t 定义*

2.3.18 视频编码配置

```
typedef struct tagVideoCodecsCfg

{

    int         iStreamIdx;

    int         videoCodec;

    int         encQuality;

    int         rateControlType;

    int         bitrate;

    int         frameRate;

    int         keyFrameInterval;

    int         StreamQuality;

} VideoCodecCfg_t;
```

成员	描述	备注
iStreamIdx	视频流索引	
videoCodec	视频编码格式	
encQuality	编码质量	
rateControlType	码率控制方式	
bitrate	比特率	
frameRate	帧率	

keyFrameInterval	关键帧间隔	
StreamQuality	流配置	

2.3.19 JPEG 编码配置

```
typedef struct tagImgCfg
{
   int  imgQuality;
   int  imgframerate;
   int  interval;
} ImgCfg_t;
```

成员	描述	备注
imgQuality	图像质量	范围 1-7，值越大图像质量越好
imgframerate	图像帧率	范围 1-5，单位为 FPS
interval	抓拍间隔	范围 1-600，单位为秒

2.3.20 音频编码配置

```
typedef struct tagAudioCfg
{
   int audioSampleRate;
   int audioCodeType;
   int audioBitRate;
   int audioIn;
```

```
} AudioCfg_t;
```

成员	描述	备注
audioSampleRate	音频采样率	
audioCodeType	音频编码格式	
audioBitRate	输出比特率	
audioIn	音频输入模式	0: line in 1: Mic in

2.3.21 移动侦测配置

```
typedef struct

{

   MotionDetection_t md;

   NET_ALARM_PARAM   alarmParam;

} NET_ALARM_MD_CFG;
```

成员	描述	备注
md	移动侦测配置参数	具体参见 MotionDetection_t 结构描述
alarmParam	移动侦测报警上报规则	具体参见 NET_ALARM_PARAM 结构描述

```
typedef struct tagMotionDetection

{

   int       enable;

   int       sensitivity;
```

```
    int       regions;

    Rectangle_t rect[MAX_MD_AREAS];

} MotionDetection_t;
```

成员	描述	备注
enable	移动侦测开关，0 为关闭，1 为打开	关闭移动侦测，后续参数可忽略
sensitivity	移动侦测灵敏度	
regions	移动侦测区域数	不得大于 MAX_MD_AREAS
rect	移动侦测区域	

2.3.22 隐私遮蔽配置

```
typedef struct tagPrivacyMask

{

    int       enable;

    int       width;

    int       height;

    int       regions;

    Rectangle_t rect[MAX_PRIVACYMASK_AREAS];

} PrivacyMask_t;
```

成员	描述	备注
enable	隐私遮蔽开关，0 为关闭，1 为打开	
width		
height		

regions	隐私遮蔽区域数	不得大于 MAX_PRIVACYMASK_AREAS
rect	隐私遮蔽区域	

2.3.23 串口配置

```
typedef struct tagSerialPortCfg
{
    int          iSerialPortType;
    int          iEnabled;
    int          iBaudrate;
    unsigned char   databits;
    unsigned char   parity;
    unsigned char   stopbits;
    unsigned char   flowctl;
} SerialPortCfg_t;
```

成员	描述	备注
iSerialPortType	串口类型	
iEnabled	启用开关，0 为关，1 为开	
iBaudrate	波特率	
databits	数据位	
parity	奇偶校验位	
stopbits	停止位	
flowctl	流控	

2.3.24 预览参数

```
typedef struct tagRealplay

{

   LONG    lChannel;

   LONG    lLinkMode;

   LONG    lStreamidx;

   BOOL    bWithaudio;

   HWND    hPlayWnd;

   char    *szMultiCastIP;

}REALPLAY;
```

成员	描述	备注
lChannel	通道号	IPC 设备可忽略
lLinkMode	连接模式	
lStreamidx	流索引	
bWithaudio	是否传输音频	
hPlayWnd	播放窗口句柄	
szMultiCastIP	多播地址	

2.3.25 协议信息

```
typedef struct tagNetVsProtocal

{

   eVSNetProtocal protoType;
```

```
    char        szDescribe[MAX_DESCRIPTION_LEN + 1];

} NetVsProtocal_t;
```

成员	描述	备注
protoType	协议类型	具体见 eVSNetProtocal
szDescribe	协议信息描述	可为空

2.3.26 协议信息列表

```
typedef struct tagNetProtoList

{

    int         count;

    NetVsProtocal_t protos[MAX_NET_PROTO_NUM];

} NetProtoList_t;
```

成员	描述	备注
count	协议个数	不能超过 MAX_NET_PROTO_NUM
protos	协议信息数组	见 NetVsProtocal_t

2.3.27 视频媒体信息

```
typedef struct tagVidMediaInfo

{

    eVideoCodecType vidType;        ///< see eVideoCodecType

    int         width;          ///< video width

    int         height;         ///< video height

    int         fps;            ///< [1-25]
```

```
    int         len;        ///< video meta data length(for h.264, sps and pps; for mp4v, vol)
    char        pData[512];    ///<
} VidMediaInfo_t;
```

成员	描述	备注
vidType	视频编码格式	见 eVideoCodecType
width	视频宽	
height	视频高	
fps	帧率	
len	视频头信息长度	
pData	视频头信息	初始化视频解码器的必要信息

2.3.28 视频全局配置

摄像机可同时输出三路视频，但是每一路的采集分辨率和视频编码模式无法单独设置。视频采集模式为全局设置，规定了 3 路视频的采集分辨率；视频编码模式规定了主次码流的视频编码格式，主次码流的编码格式必须一致。

```
typedef struct tagVideoCaptureCfg
{
    int captureMode;
    int SubcaptureMode;
} VideoCaptureCfg_t;
```

成员	描述	备注
captureMode	视频编码模式	主码流视频分辨率
SubcaptureMode	视频编码模式	子码流视频分辨率

2.3.29 本地视频输出配置

```
typedef struct tagVideoOutputCfg

{

   int EnableBNC;

} VideoOutputCfg_t;
```

成员	描述	备注
EnableBNC	BNC 输出开关，0 为关闭，1 为打开	

2.3.30 音频媒体信息

```
typedef struct tagAudMediaInfo

{

   eAudioCodecType audType;

   int          samplerate;

   int          bitrate;

   int          channels;

   int          len;

   char          pData[512];

} AudMediaInfo_t;
```

成员	描述	备注
audType	音频编码格式	
samplerate	采样率	目前仅支持 8Khz 采样率
bitrate	比特率	单位为 bps

channels	声道	
len	音频信息长度	
pData	音频信息	

2.3.31 视频遮挡配置

```
typedef struct

{

   CameraTamper_t    ct;

   NET_ALARM_PARAM   alarmParam;

} NET_ALARM_CAMTAM_CFG;
```

成员	描述	备注
ct	视频遮挡配置参数	见 CameraTamper_t 结构定义
alarmParam	视频遮挡报警上报规则	具体参见 NET_ALARM_PARAM 结构描述

```
typedef struct tagCameraTamper

{

   int enable;

   int duration;

   int sensitivity;

} CameraTamper_t;
```

成员	描述	备注

enable	使能开关	
duration	报警时延	
sensitivity	灵敏度	

2.3.32 SMTP 服务器配置

```
typedef struct tagSMTPServer

{

   char        szUserName[MAX_USERNAME_LEN + 1];

   char        szPassword[MAX_PWD_LEN + 1];

   char        szSMTPServer[MAX_DOMAIN_NAME + 1];

   unsigned short smtpport;

   int         smtpAuth;

} SMTPServer_t;
```

成员	描述	备注
szUserName	服务器登录名	
szPassword	服务器登录密码	
szSMTPServer	SMTP 服务器域名	
smtpport	SMTP 服务端口	
smtpAuth	SMTP 登录认证	

2.3.33 报警布防参数

```
typedef struct
```

```
{
    DWORD    dwSize;
    BYTE     byLevel;
    BYTE     byRes[15];
} NET_SETUPALARM_PARAM;
```

成员	描述	备注
dwSize		
byLevel		
byRes		

2.3.34 报警设备信息

```
typedef struct
{
    BYTE    byUserIDValid;
    BYTE    bySerialValid;
    BYTE    byVersionValid;
    BYTE    byDeviceNameValid;
    BYTE    byMacAddrValid;
    BYTE    byLinkPortValid;
    BYTE    byDeviceIPValid;
    BYTE    bySocketIPValid;
    LONG    lUserID;
    BYTE    sSerialNumber[MAX_SERIAL_NUM + 1];
```

```
    DWORD    dwDeviceVersion;
    char     sDeviceName[MAX_DEVICE_NAME_LEN];
    BYTE     byMacAddr[MAC_LEN];
    WORD     wLinkPort;
    char     sDeviceIP[128];
    char     sSocketIP[128];
    BYTE     byIpProtocol;
    BYTE     byRes2[11];
} NET_ALARMER;
```

成员	描述	备注
byUserIDValid		未使用
bySerialValid		未使用
byVersionValid		未使用
byDeviceNameValid		未使用
byMacAddrValid		未使用
byLinkPortValid		未使用
byDeviceIPValid		未使用
bySocketIPValid		未使用
lUserID	IPC 登陆 ID，即 NET_Login 函数返回的值。	
sSerialNumber		未使用
dwDeviceVersion		未使用
sDeviceName		未使用

byMacAddr		未使用
wLinkPort		未使用
sDeviceIP		未使用
sSocketIP		未使用
byIpProtocol		未使用

2.3.35 普通报警信息

```
typedef struct
{
    DWORD    dwAlarmType;
    DWORD    dwAlarmInputNumber;
    DWORD    dwAlarmOutputNumber[MAX_ALARMOUT];
    DWORD    dwAlarmRelateChannel[MAX_CHANNUM];
    DWORD    dwChannel[MAX_CHANNUM];
    DWORD    dwDiskNumber[MAX_DISKNUM];
} ALARMINFO_COMMON, *PALARMINFO_COMMON;
```

成员	描述	备注
dwAlarmType	报警类型： 0 - 信号量报警；1 - 硬盘满； 3 - 移动侦测； 4 - 硬盘未格式化； 5 - 读写硬盘出错； 6 - 遮挡报警。	
dwAlarmInputNumber	报警输入端口。	
dwAlarmOutputNumber	触发的报警输出端口。当报警类型为信号量报警时，值为 1 表示该报警端口输出。	
dwAlarmRelateChannel	触发的录像通道。当报警类型为信号量报警	

	时，值为 1 表示该通道录像，如 dwAlarmRelateChannel[0]表示触发第 1 个通道录像。	
dwChannel	发生报警的通道。当报警类型为 3，6 时有效，如 dwChannel[0]值为 1 表示第 1 个通道报警。	
dwDiskNumber	发生报警的硬盘。当报警类型为 1，4，5 时有效，dwDiskNumber[0]值为 1 表示 1 号硬盘异常	

2.3.36 视频帧头

```
typedef struct
{
    unsigned short usFrameType;
    unsigned long  ulSize;
    unsigned long  ulFrameSeqNum;
    unsigned long  ulTimeStamp;
    long long      llSysTimestamp;
    unsigned long  ulEncryptSize;
} NET_FRAME_HEAD;
```

成员	描述	备注
usFrameType	帧类型	
ulSize	帧长	
ulFrameSeqNum	帧序号	
ulTimeStamp	媒体时间戳，从 0 开始计算	

llSysTimestamp	设备系统时间戳	
ulEncryptSize	加密长度	只适用于加密机型

2.3.37 服务端口参数

```
typedef struct tagServerPorts

{

   short unsigned int httpPort;

   short unsigned int httpsPort;

   short unsigned int ftpSrvPort;

   short unsigned int rtspPort;

   short unsigned int tcpPort;

} ServerPorts_t;
```

成员	描述	备注
httpPort	HTTP 端口，默认为 80	
httpsPort	HTTPS 端口，默认为 81	
ftpSrvPort	ftp 服务端口，默认为 21	
rtspPort	rtsp 端口，默认为 554	
tcpPort	tcp 端口，默认为 6000	

【注意】除默认端口外，端口可用范围为 10000-50000。

2.3.38 录像参数

```
typedef struct

{

   TimeTable_t   struSched;
```

```
    unsigned int  uPreRecordLen;

    unsigned int  uRedundancyLen;

    unsigned int  uRecordType;

} NET_RECORD_CFG;
```

成员	描述	备注
struSched	录像时间表	
uPreRecordLen	预录长度，0 为不预录	
uRedundancyLen	延像长度	
uRecordType	录像类型	

2.3.39 抓拍参数

```
typedef struct

{

TimeTable_t struSched;

    unsigned int uQulity;

    unsigned int uFrameRate;

unsigned int uIntervalTime;

} NET_SNAP_CFG;
```

成员	描述	备注
struSched	抓拍计划表	
uQulity	抓拍图像质量	
uFrameRate	帧率	

uIntervalTime	抓拍间隔	

2.3.40 上传配置参数

```
typedef struct
{
   unsigned int uUploadEnable;
   unsigned int uUploadType;
} NET_UPLOAD_CFG;
```

成员	描述	备注
uUploadEnable	抓拍计划表	
uUploadType	抓拍图像质量	

2.3.41 SD 卡参数

```
typedef struct
{
   unsigned int uOverWrite;
   unsigned int uAutoFormat;
   unsigned int uSDStatus;
   unsigned int uSDVolume;
   unsigned int uSDFreeSpace;
} NET_SDCARD_PARAM;
```

成员	描述	备注
uOverWrite	1 为录满覆写，0 为录满停止录像	
uAutoFormat	插入未知格式 SD 卡是否自动格式化，0 为否，1 为是	
uSDStatus	SD 卡状态	
uSDVolume	SD 卡容量	
uSDFreeSpace	SD 卡剩余容量	

2.3.42 IO 输出配置

```
typedef struct
{
    unsigned int uDefaultStatus;
   unsigned int uIOOutStatus;
unsigned int uTimeDelay;
   unsigned int uTimePluse;
   unsigned int uFreqMulti;
   unsigned int uDutyRate;
} NET_IO_OUTCFG;
```

成员	描述	备注
uDefaultStatus	IO 默认状态	
uIOOutStatus	IO 起效时状态	
uTimeDelay	IO 有效持续时间	
uTimePluse	脉冲间隔时间	脉冲模式下有效

uFreqMulti	倍频	脉冲模式下有效
uDutyRate	占空比	脉冲模式下有效

2.3.43 报警处理

```
typedef struct

{

   unsigned int uActionMask;

char  byRelAlarmOut[MAX_IOOUT_NUM];

unsigned int uDuration;

} NET_ALARM_HANDLE;
```

成员	描述	备注
uActionMask	报警处理动作，可多种处理并存	见 eAlarmOutType 枚举的定义
byRelAlarmOut	报警触发的输出通道，0-不触发，1-触发输出	
uDuration	报警持续时间	

2.3.44 报警输入

```
typedef struct

{

   char  szAlarmInName[MAX_ALARMIN_NAME + 1];

   unsigned int uAlarmType;

   unsigned int uAlarmEnable;

   NET_ALARM_HANDLE handle;
```

```
} NET_ALARMIN_CFG;
```

成员	描述	备注
szAlarmInName	报警输入名称	
uAlarmType	报警输入类型	
uAlarmEnable	报警输入使能开关	
handle	报警处理方式	

2.3.45 DNS 配置

```
typedef struct tagDNSAddr

{

   IPAddr_t  DNS1;

   IPAddr_t  DNS2;

} DNSAddr_t;
```

2.3.46 PPPoE 配置

```
typedef struct tagPPPOECfg

{

   int      enable;

   char     szUserName[MAX_USERNAME_LEN + 1];

   char     szPassword[MAX_PWD_LEN + 1];

   IPAddr_t struPPPoEIP;

} PPPOECfg_t;
```

2.3.47 DDNS 设置

```
typedef struct tagDDNSCfg

{

   int  bEnableDDNS;

   int  hostIndex;

   char szUserName[MAX_USERNAME_LEN + 1];

   char szPassword[MAX_PWD_LEN + 1];

char szDomainName[MAX_DOMAIN_NAME + 1];

int  port;

int  updatePerTime;

} DDNSCfg_t;
```

成员	描述	备注
bEnableDDNS	使能开关	
hostIndex	服务器类型	见 eDDNSServer
szUserName	登录用户名	
szPassword	登录密码	
szDomainName	服务器域名	
port	服务端口	
updatePerTime	更新时间	

2.3.48 以太网配置

```
typedef struct tagEtherlink

{
```

```
    IPAddr_t    struIP;

    IPAddr_t    struIPMask;

    IPAddr_t    struGatewayIpAddr;

    char        byMAC[MAC_LEN + 1];

} Etherlink_t;
```

成员	描述	备注
struIP	IP 地址	
struIPMask	子网掩码	
struGatewayIpAddr	网关	
byMAC	MAC 地址	

2.3.49 基本网络配置

```
typedef struct tagNetCfg

{

    int         bUseDhcp;

    Etherlink_t   struEtherNet;

    DNSAddr_t     struDNSServer;

    PPPOECfg_t    struPPPoE;

} NetCfg_t;
```

成员	描述	备注
bUseDhcp	是否启用 DHCP	

struEtherNet	以太网设置	
struDNSServer	DNS 服务器设置	
struPPPoE	PPPoE 设置	

2.3.50 WEP 加密

```
typedef struct tagWEP

{

   int  authenticationType; ///< 0 for open, 1 for share, 2 for auto.

   int  keyIndex;

   int  keyLength;

   char encryption[WIFI_WEP_MAX_KEY_COUNT][127 + 1];///< WEPKEYLIST_LEN

} WEP_t;
```

成员	描述	备注
authenticationType	权限类型	
keyIndex	激活密钥索引	
keyLength	密钥长度，64 或 128	
encryption	密钥信息	

2.3.51 无线网络配置

```
typedef struct tagWIFICfg

{

   int          enable;              ///< 0:disable

   char                    ssid[MAX_SSID_LEN + 1]; ///<
```

```
    int         authEnc;            ///< see eAuthEnc
    char                pwd[MAX_WIFIKEY_LEN + 1];///<
    IPAddress_t         ip;
} WIFICfg_t;
```

成员	描述	备注
enabled	WIFI 使能开关，0 为禁用	未使用
ssid	Wifi 连接的 AP 的 essid	
authEnc	AP 的加密模式	查看 eAuthEnc 定义
pwd	密码	
ssid	SSID	
IPAddress_t	Wifi 网络参数配置，可使用 dhcp 或静态 IP 模式	

2.3.52 报警信息

```
typedef struct tagAlaramInfo
{
    LONG        lCommand;
    NET_ALARMER na;
    char*           pAlarmInfo;
    unsigned int        dwBufLen;
}ALARM_INFO,*LPALARM_INFO;
```

成员	描述	备注
lCommand	上传的报警类型，每个 lCommand 类型	类型及其对应结构可

	对应一个报警信息 pAlarmInfo 的机构	见下表
na	报警设备信息	详见 NET_ALARMER 结构定义
pAlarmInfo	报警信息每个 lCommand 类型对应一个报警信息 pAlarmInfo 的机构	类型及其对应结构可见下表
dwBufLen	报警信息缓存长度	

报警类型	值	含义	输出信息结构
COMMAND_ALARM_COMMON	0x01	普通设备报警信息	ALARMINFO_COMMON

2.3.53 报警事件

```
typedef struct tagAlarmEventCfg
{
    int                                 type;
    int                                 mode;
    int                                 duration;
    TimeTable_t         sche;
    AlarmOutputCfg_toutput;
}AlarmEventCfg_t;
```

成员	描述	备注
type	报警类型	
mode	报警事件监测模式	0 关闭，7*24 小时，2 定时
duration	报警延时	【1,300】

sche	计划时间表	参见 TimeTable_t
output	报警处理设置	参见 AlarmOutPutCgf_t

2.3.54 异常信息

```
typedef struct tagExceptionInfo

{

   unsigned long uExceptionType;

   HANDLE hIPC;

   HANDLE hChannel;

}EXCEPTION_INFO, *PEXCEPTION_INFO;
```

成员	描述	备注
uExceptionType	异常类型	异常类型下表
hIPC	发生异常设备登录句柄	
hChannel	发生异常设备播放句柄	

异常类型	值	说明
EXCEPTION_EXCHANGE	0x8000	与 IPC 连接断开
EXCEPTION_RECONNECT	0x8005	与 IPC 连接断开后，重连成功

2.3.55 播放参数

```
typedef struct

{
```

```
int nChannel;

DEFAVDATAPROC lpProc;

void* lpParam;

HWND hPlayWnd;

}OPEN_CHANNEL_INFO,*LPOPEN_CHANNEL_INFO;
```

成员	描述	备注
nChannel	流索引	
lpProc	码流数据回调函数，可设为 NULL	
lpParam	码流数据回调函数用户数据	
hPlayWnd	播放窗口句柄。可设为 NULL，这时播放函数将不处理解码及窗口播放	

2.3.56 PTZ 参数

```
typedef struct tagPtzCfg

{

unsigned char speed;

unsigned char iris;

unsigned char light;

unsigned char wiper;

unsigned char zmode;

PtzPreset_t                    preset[MAX_PTZPRESET_NUM];

}PtzCfg_t;
```

成员	描述	备注
speed	移动的速度	有效范围[0, 7]，值越大，速度越快
iris	光圈	0:关闭， 1：打开
light	灯光	0:关闭， 1：打开
wiper	雨刷	0:关闭， 1：打开
zmode	聚焦模式	PTZ_ZOOM_MODE_AUTO:自模式, PTZ_ZOOM_MODE_MANUAL：手动模式 PTZ_ZOOM_MODE_TRIGGER：触发模式
PtzPreset	预置位，最多支持MAX_PTZPRESET_NUM	

2.3.57 PTZ 控制参数

```
typedef struct tagPtzCtrl

{

int             protocol;

int             cmd;

usigned char            addr

union

{

struct

{

unsigned char           hdir;     //0:left 1:right
```

```
unsigned char            vdir;      //0:up 1:down

    }move;

    struct

{

    unsigned char       dir;        //0:wide  1:tele

}zoom;

struct

{

    unsigned char       dir;        //0:far   1:near

}focus;

struct

{

unsigned char               dir; //0:close 1:open

}iris;

    struct

    {

    unsigned char       action; //0:go 1:add 2:del

    unsigned char       id;

}preset;

unsigned char                       args[24];

};

} PtzCtrl_t;
```

成员	描述	备注
protocol	Ptz 协议类型，参见 PTZ_PROTO_XX XX	目前只支持 PTZ_PROTO_PELCOD 与 PTZ_PROTO_ZXAF
cmd	控制命令，可组合，参见 PTZ_XXXX	
addr	地址，目前固定为 0x0	
move	移动命令时，所使用参数	
zoom	变倍命令时，所使用参数	
focus	聚焦命令时，所使用参数	
iris	光圈命令时，所使用参数	
preset	预置位命令时，所使用参数	

2.3.58 WIFI 参数

```
typedef struct tagWIFICfg
{
    int         enable;
    char                ssid[MAX_SSID_LEN + 1];
    int         authEnc;
    char                pwd[MAX_WIFIKEY_LEN + 1];
    IPAddress_t         ip;
} WIFICfg_t;
```

成员	描述	备注

Enable	WIFI 使能开关（暂不可用）	
Ssid	WIFI 连接账号	
Authenc	WIFI 加密方式	
Pwd	WIFI 连接密码	
IP	IP 地址信息	参见 IPAddress_t 结构

2.4 接口

2.4.1 SDK 初始化

2.4.1.1 NETSDK_Init

【函数原型】*void NETSDK_Init()*

【功能】完成 Net SDK 初始化。此函数是调用 Net SDK 其它接口的必要前提。

【参数】无。

【返回值】无。

【声明头文件】IPCNetSDK_Interface.h

【使用方法】同一进程可多次调用此接口，但只有第一次调用会实际做初始化工作，后续的调用增加SDK引用计数。

2.4.1.2 NETSDK_Cleanup

【函数原型】*void NETSDK_Cleanup()*

【功能】去初始化，释放分配的内存等资源。

【参数】无。

【返回值】无。

【声明头文件】IPCNetSDK_Interface.h

【使用方法】 在退出程序前调用此函数。NETSDK_Cleanup 调用的次数要和初始化调用 NETSDK_Init 的次数一致，才能完成去初始化工作。

2.4.2 IPC 注册

2.4.2.1 NETSDK_LoginIPC

【函数原型】*HANDLE NETSDK_LoginIPC(const char* szIP, int nPort,const char* szUser, const char *szPwd, DEFIPCNETMESSAGE lpProc, void *param);*

【功能】用户注册登录设备。

【参数】

szIP

[in]设备 IP 地址。

nPort

[in]设备登陆端口。

szUser

[in]登陆用户名。

szPwd

[in]登陆用户密码。

lpProc

[in] 接收连接断开异常、重连接成功、报警的的回调函数地址。

param

[in]用户数据。

【回调函数】

```
typedef void (__stdcall* DEFIPCNETMESSAGE)(

HANDLE hIPC,

DWORD dwMsg,

void* lpData,
```

```
int nLength,

void *param);
```

【回调函数参数】

hIPC

[out] 异常或上报告警 IPC 的用户句柄，是 NETSDK_LoginIPC 的返回值

dwMsg

[out] 回调类型，包括异常上报及告警，详见枚举 eInfoType 的定义

lpData

[out] 回调信息（异常或告警）信息内容，不同的异常类型对应不同的结构，如下：

回调类型	回调类型定义枚举值	对应回调信息结构
告警	INFOTYPE_ALARM	ALARM_INFO
异常	INFOTYEP_EXCEPTION	EXCEPTION_INFO

lnLength

[out] 异常或告警信息内容长度，为相应信息机构的长度。

param

[out] 用户数据

【返回值】NULL表示失败，其他值表示返回的用户句柄。该用户句柄具有唯一性，后续对设备的操作都需要通过此句柄实现。获取错误码调用 NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_CREATTHREADFAIL	0x20000009	试图创建线程失败

NET_SENDFAIL	0x20000006	发送数据失败
NET_RECVFAIL	0x20000007	接收数据失败
NET_WRONGDATALEN	0x2000000b	接收数据的长度错误
NET_WRONGDATA	0x2000000c	接收数据格式错误或与期望不符
NET_CREATESOCKETFAIL	0x20000008	创建套接字失败
TNET_CONNECTFAIL	0x20000005	网络连接失败
NET_IPC_BAD_HEADLINE	0x20002000	IPC反馈，请求的用户名或密码、序列号为空
NET_IPC_NOSUPPORT	0x20002001	IPC反馈，未知的请求
NET_IPC_NOCONTENT	0x20002002	IPC反馈，请求内容为空
NET_IPC_ERRORDATA	0x20002003	IPC反馈，数据错误
NET_IPC_MAX_CONNECTION	0x20002005	IPC反馈，IPC连接已满
NET_IPC_PRIVILEGE_INVALID	0x20002006	IPC反馈，用户权限不足
NET_IPC_INVALID_USER	0x20002007	IPC反馈，无效的用户

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.2.2 NETSDK_LogoutIPC

【函数原型】*void NETSDK_LogoutIPC(HANDLE hIPC)*

【功能】用户注销

【参数】

hIPC

[in] 用户句柄，*NETSDK_LoginIPC* 的返回值

【返回值】无。

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.3 实时预览

2.4.3.1 NETSDK_OpenChannel

【函数原型】*HANDLE NETSDK_OpenChannel(HANDLE hIPC,LPOPEN_CHANNEL_INFO lpInfo)*

【功能】打开实时预览。若输入播放窗口句柄有效，接口会导入播放 SDK 动态连接库中的接口函数，解码并在输入播放窗口上显示图像。如数据处理回调函数不为空，还可从网络接收数据自行解码和处理。

【参数】

hIPC

[in] NETSDK_LoginIPC 的返回值

lpInfo

[in] 预览参数，详见 OPEN_CHANNEL_INFO 结构定义

【返回值】失败返回NULL，成功返回实时预览句柄，将作为相关函数的参数。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_LOADPLAYDLLFAIL	0x2000000f	加载播放动态库失败
NET_CREATTHREADFAIL	0x20000009	试图创建线程失败
NET_NOCONNECTED	0x2000000d	与IPC连接断开

NET_CREATESOCKETFAIL	0x20000008	创建套接字失败
NET_CONNECTFAIL	0x20000005	网络连接失败
NET_SENDFAIL	0x20000006	发送数据失败
NET_RECVFAIL	0x20000007	接收数据失败
NET_WRONGDATA	0x2000000c	接收数据格式错误或与期望不符
NET_INITPLAYFAIL	0x20000010	播放初始化失败
NET_IPC_VIDIO_NOSOURCE	0x20000193	IPC反馈，IPC播放数已满
NET_IPC_VIDIO_SENSERERR	0x20000194	IPC反馈，IPC的Senser错误

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.3.2 NETSDK_CloseChannel

【函数原型】void NETSDK_CloseChannel(HANDLE hChannel)

【功能】停止预览

【参数】

hChannel

[in] NETSDK_OpenChannel 返回值

【返回值】无。

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.3.3 NETSDK_PlaySound

【函数原型】BOOL NETSDK_PlaySound(HANDLE hChannel)

【功能】播放声音

【参数】

hChannel

[in] NETSDK_OpenChannel 返回值

【返回值】TRUE表示成功；FALSE表示失败。获取错误码调用 NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_LOADPLAYDLLFAIL	0x2000000f	加载播放动态库失败
NET_UNINITPLAY	0x20000012	播放动态库未初始化。调用NETSDK_OpenChannel函数预览时，如参数未带窗口句柄，播放动态库就不会初始化
NET_PLAYSOUNDFAIL	0x20000013	播放解码库打开声音失败

【注意】本函数只对调用 NETSDK_OpenChannel 函数预览时，参数带窗口句柄的播放有效，而对通过解码回调函数自行解码的播放不进行控制。

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.3.4 NETSDK_PauseSound

函数原型】BOOL NETSDK_PauseSound(HANDLE hChannel)

【功能】关闭声音

【参数】

hChannel

[in] NETSDK_OpenChannel 返回值

【返回值】TRUE 表示成功；FALSE 表示失败。获取错误码调用 NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_PAUSESOUNDFAIL	0x20000014	播放解码库关闭声音失败
NET_LOADPLAYDLLFAIL	0x2000000f	加载播放动态库失败
NET_UNINITPLAY	0x20000012	播放动态库未初始化.调用NETSDK_OpenChannel函数预览时，如参数未带窗口句柄，播放动态库就不会初始化

【注意】本函数只对调用 NETSDK_OpenChannel 函数预览时，参数带窗口句柄的播放有效，而对通过解码回调函数自行解码的播放不进行控制。

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.3.5 NETSDK_CaptureBMP

*【函数原型】BOOL NETSDK_CaptureBMP(HANDLE hChannel, char *szPath)*

【功能】预览时，单帧数据捕获并保存成 BMP 图片

【参数】

hChannel

[in] NETSDK_OpenChannel 返回值

szPath

[in] 保存图象的文件路径及文件名

【返回值】TRUE表示成功；FALSE表示失败。获取错误码调用 NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_LOADPLAYDLLFAIL	0x2000000f	加载播放动态库失败

NET_UNINITPLAY	0x20000012	播放动态库未初始化.调用NETSDK_OpenChannel函数预览时，如参数未带窗口句柄，播放动态库就不会初始化
NET_TF_NET_CAPTUREFAIL	0x20000011	播放解码库抓图失败

【注意】本函数只在对调用 NETSDK_OpenChannel 函数预览时，参数带窗口句柄的播放的抓图才有效。

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.3.6 NETSDK_GetHeadData

【函数原型】char NETSDK_GetHeadData(HANDLE hChannel, int* nLength)*

【功能】预览时，获取视频/音频流信息

【参数】

hChannel

[in] NETSDK_OpenChannel 返回值

nLength

[out] 返回的视频/音频流信息缓存长度

【返回值】返回视频流信息缓存指针，详见结构VidMediaInfo_t，根据nLength返回的长度，可判断缓存中是否包含音频流信息，如果包含，音频信息紧跟视频流之后，音频信息详见结构AudMediaInfo_t。返回NULL表示获取失败。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.3.7 NETSDK_GetJPEG

*【函数原型】BOOL NETSDK_GetJPEG(HANDLE hIPC, char *szFilePath, DEFPICTUREDATAPROC pProc, void *param)*

【功能】对已经登陆的 IPC，拍照并保存成 JPEG 图片

【参数】

hIPC

[in] NETSDK_LoginIPC 返回值

szFilePath

[in] 保存图象的文件路径及文件名。可为 NULL，表示不由函数保存图片。

pProc

[in] 接收拍照图片的流数据的回调函数。可为 NULL，表示不接收处理拍照图片的流数据

param

[in] 用户数据

【回调函数】

```
typedef void (__stdcall* DEFPICTUREDATAPROC)(HANDLE hIPC,
BYTE *lpData,
DWORD nLength,
BYTE *extra,
DWORD extrelen,
void *param);
```

【回调函数参数】

hIPC

[out] IPC 的登陆句柄，是 NETSDK_LoginIPC 的返回值

lpData

[out] IPC 拍照 JPEG 流数据缓存指针

nLength

[int] IPC 拍照 JPEG 流数据缓存长度

extra

[int] 附加数据缓存指针，目前未使用

extrelen

[int] 附加数据缓存长度，目前未使用

param

[int] 用户数据

【返回值】TRUE表示成功；FALSE表示失败。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_NOCONNECTED	0x2000000d	与IPC连接断开
NET_CREATESOCKETFAIL	0x20000008	创建套接字失败
NET_CONNECTFAIL	0x20000005	网络连接失败
NET_SENDFAIL	0x20000006	发送数据失败
NET_RECVFAIL	0x20000007	接收数据失败
NET_WRONGDATA	0x2000000c	接收数据格式错误或与期望不符

【注意】本函数只在对调用 NETSDK_OpenChannel 函数预览时，参数带窗口句柄的播放的抓图才有效。

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.3.8 NETSDK_GetVideoStatics

*【函数原型】BOOL NETSDK_GetVideoStatics(HANDLE hChannel, float *fps, float *bps);*

【功能】获取实时流统计的帧率和比特率

【参数】

hChannel

[in] NETSDK_OpenChannel 返回值

fps

[out]获取的帧率

bps

[out] 获取的比特率

【返回值】TRUE 表示成功；FALSE 表示失败。获取错误码调用 NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误

【注意】如获取到的帧率或比特率为 0，可稍后再次获取。

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.4 远程参数配置

2.4.4.1 NETSDK_GetParam

【函数原型】*NET_EXPROT BOOL NETSDK_GetParam(HANDLE hIPC,int nCMD, LONG cmdType, int nChIndex, int nStreamIndex, void* lpData, int* pnLength)*

【功能】获取设备配置信息。本接口为同步调用，调用后阻塞，直到收到设备应答方能返回。若设备无应答，则等待超时返回。

【参数】

hIPC

[in] IPC 登陆句柄，NETSDK_LoginIPC 返回值

nCMD

[in]配置命令，不同的命令会传出不同的配置信息结构，见下表：

cmd宏定义	含义	对应结构体	值
CMD_GET_NTP	时间配置	NtpCfg_t	
CMD_GET_DEVICEINFO	设备信息	DeviceInfo_t	
CMD_GET_DEVICESTATUS	设备运行状态	DeviceStatus_t	
CMD_GET_TIME	设备时间	TimeInfo_t	
CMD_GET_FTPCFG	FTP 配置	FTPCfg_t	
CMD_GET_MJPEGCFG	MJPEG 配置	ImgCfg_t	
CMD_GET_CAMERACFG	摄像头前端配置	CameraCfg_t	
CMD_GET_MOTIONDETECT	移动侦测配置	NET_ALARM_MD_CFG	
CMD_GET_CAMERATAMPER	视频遮挡配置	NET_ALARM_CAMTAM_CFG	
CMD_GET_PPPOE	PPPOE 配置		
CMD_GET_SMTPCFG	Email 配置	MailCfg_t	
CMD_GET_DDNSCFG	DDNS 配置	DDNSCfg_t	
CMD_GET_VIDEOCODECCFG	视频编码	VideoCodecCfg_t	
CMD_GET_AUDIOCODECCFG	音频编码	AudioCfg_t	
CMD_GET_IPFILTERCFG	IP 地址过滤配置	IPFilter_t	
CMD_GET_VIDEOEFFECT	图像效果配置		
CMD_GET_PTZPROTOCOL	获取 PTZ 协议		

CMD_GET_SERVERPORTS	服务端口配置	ServerPorts_t	
CMD_GET_STORAGEDEVINFO	存储信息		
CMD_GET_DSTCFG	夏时制设置	SummerTime_t	
CMD_GET_MULTICAST_CFG	多播配置		
CMD_GET_OSDCFG	OSD 配置	OSDCfg_t	
CMD_GET_USERLIST	用户信息列表	UserList_t	
CMD_GET_LOGCFG	日志配置		
CMD_GET_REC_PACKET_CFG	录像打包配置		
CMD_GET_RECORD_CFG	定时录像配置	NET_RECORD_CFG	
CMD_GET_VIDEOGLOBALCFG	视频全局设置	VideoGlobalCfg_t	
CMD_GET_VIDEOOUTPUTCFG	视频输出设置	VideoOutputCfg_t	
CMD_GET_VSPROTOCAL	IPC 协议	NetProtoList_t	
CMD_GET_PTZCFG	Ptz 参数	PtzCfg_t	
CMD_GET_WIFI	WIFI 连接信息	WIFICfg_t	
CMD_GET_NETLINKINFO	网络信息	NetLinkInfo_t	
CMD_GET_ALARMEVTCFG	报警事件参数	AlarmEventCfg_t	

cmdType

[in] 获取配置的方式，详见枚举 eGetPara

nChIndex

[in] 通道号，目前只有一个通道号0

nStreamIndex

[in] 码流索引，目前未使用

lpData

[out] 获取到的配置信息缓存

pnLength

[out] 获取到的配置信息缓存长度

【返回值】TRUE表示成功；FALSE表示失败。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_SENDFAIL	0x20000006	发送数据失败
NET_WAITTIMEOUT	0x20000002	接收配置数据超时
NET_IPC_BAD_HEADLINE	0x20002000	IPC反馈，请求的用户名或密码、序列号为空
NET_IPC_NOSUPPORT	0x20002001	IPC反馈，未知的请求
NET_IPC_NOCONTENT	0x20002002	IPC反馈，请求内容为空
NET_IPC_ERRORDATA	0x20002003	IPC反馈，数据错误
NET_IPC_INVALID_CMD	0x20002004	IPC反馈，无效的命令
NET_IPC_PRIVILEGE_INVALID	0x20002006	IPC反馈，用户权限不足

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.4.2 NETSDK_SetParam

【函数原型】*BOOL NETSDK_SetParam(HANDLE hIPC,int nCMD, LONG cmdType, int nChIndex, void* lpData, int nLength)*

【功能】设置设备的配置信息。本接口为同步调用，调用后阻塞，直到收到设备应答方能返回。若设备无应答，则等待超时返回。

【参数】

hIPC

[in] IPC 登陆句柄，NETSDK_LoginIPC 返回值

nCMD

[in]配置命令，不同的命令应传入不同的配置信息结构，见下表：

cmd宏定义	含义	对应结构体
CMD_SET_NTP	NTP 配置	NtpCfg_t
CMD_SET_TIME	设备时间	TimeInfo_t
CMD_SET_FTPCFG	FTP 配置	FTPCfg_t
CMD_SET_MJPEGCFG	MJPEG 配置	ImgCfg_t
CMD_SET_CAMERACFG	摄像头前端配置	CameraCfg_t
CMD_SET_PRIVACYMASK	隐私遮蔽配置	PrivacyMask_t
CMD_SET_MOTIONDETECT	移动侦测配置	NET_ALARM_MD_CFG
CMD_SET_CAMERATAMPER	视频遮挡配置	NET_ALARM_CAMTAM_CFG
CMD_GET_SNAP_CFG	抓拍配置	NET_SNAP_CFG
CMD_GET_WIFI	WIFI 配置	NetworkInterfaceList_t
CMD_GET_IOOUTCFG	IO 输出配置	
CMD_SET_SMTPCFG	Email 配置	MailCfg_t
CMD_SET_DDNSCFG	DDNS 配置	DDNSCfg_t
CMD_SET_VIDEOCODECCFG	视频编码	VideoCodecCfg_t
CMD_SET_AUDIOCODECCFG	音频编码	AudioCfg_t
CMD_SET_IPFILTERCFG	IP 地址过滤配置	IPFilter_t

CMD_SET_VIDEOEFFECT	图像效果配置	
CMD_SET_SERVERPORTS	服务端口配置	ServerPorts_t
CMD_SET_DSTCFG	夏时制设置	SummerTime_t
CMD_SET_MULTICAST_CFG	多播配置	
CMD_SET_OSDCFG	OSD 配置	OSDCfg_t
CMD_SET_USERLIST	用户信息列表	UserList_t
CMD_SET_LOGCFG	日志配置	
CMD_SET_REC_PACKET_CFG	录像打包配置	
CMD_SET_RECORD_CFG	定时录像配置	NET_RECORD_CFG
CMD_SET_VIDEOGLOBALCFG	视频全局设置	VideoGlobalCfg_t
CMD_SET_VIDEOOUTPUTCFG	视频输出设置	VideoOutputCfg_t
CMD_SET_VSPROTOCAL	IPC 协议	NetProtoList_t
CMD_SET_PTZCFG	Ptz 参数	PtzCfg_t
CMD_SET_ALARMEVTCFG	报警事件参数	AlarmeventCfg_t
CMD_SET_WIFI	WIFI 设置	WIFICfg_t

cmdType [in] 获取配置的方式，详见枚举 eSetPara

nChIndex [in] 通道号，目前只有一个通道号0

nStreamIndex [in] 码流索引，目前未使用

lpData [in]配置信息缓存指针

pnLength [in] 配置信息缓存长度

【返回值】TRUE表示成功；FALSE表示失败。获取错误码调用 NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_SENDFAIL	0x20000006	发送数据失败
NET_WAITTIMEOUT	0x20000002	接收配置数据超时
NET_NOCONNECTED	0x2000000d	与IPC连接断开
NET_XMLCREATEORPARSEFAIL	0x2000000e	XML合成错误
NET_IPC_BAD_HEADLINE	0x20002000	IPC反馈，请求的用户名或密码、序列号为空
NET_IPC_NOSUPPORT	0x20002001	IPC反馈，未知的请求
NET_IPC_NOCONTENT	0x20002002	IPC反馈，请求内容为空
NET_IPC_ERRORDATA	0x20002003	IPC反馈，数据错误
NET_IPC_INVALID_CMD	0x20002004	IPC反馈，无效的命令
NET_IPC_PRIVILEGE_INVALID	0x20002006	IPC反馈，用户权限不足

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.5 获取设备能力集

2.4.5.1 NETSDK_GetDeviceCaps

【函数原型】*BOOL NETSDK_GetDeviceCaps(HANDLE hIPC, LONG dwCapType, unsigned long dwCapacity, char *pOutBuf, unsigned long *dwOutLength)*

【功能】获取设备能力集。

【参数】

hIPC [in] IPC 登陆句柄，NETSDK_LoginIPC 返回值

dwCapType [in] 能力类型，见下表：

cmd宏定义	值	含义
DEVICE_SYSTEM_ABILITY	0x2	系统能力集
DEVICE_NETWORK_ABILITY	0x3	网络能力集
DEVICE_ALARM_ABILITY	0x4	告警能力集
DEVICE_ENCODE_ABILITY	0x5	解码能力集

dwCapacity [in] 传入的存放能力集数据缓存的长度

pOutBuf [out] 存放能力集数据缓存

dwOutLength [out] 获取的能力集数据长度

【返回值】TRUE表示成功；FALSE表示失败。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_SENDFAIL	0x20000006	发送数据失败
NET_WAITTIMEOUT	0x20000002	接收数据超时
NET_NOCONNECTED	0x2000000d	与IPC连接断开

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.6 用户管理

2.4.6.1 NETSDK_OperateUserInfo

【函数原型】*BOOL NETSDK_OperateUserInfo(HANDLE hIPC, int nOperateType, UserList_t *pUserInfo)*

【功能】管理设备用户。

【参数】

hIPC [in] IPC 登陆句柄，NETSDK_LoginIPC 返回值

nOperateType [in] 操作用户类型，见枚举 eOperateType：

pUserInfo [in] 要操作的用户信息

【返回值】TRUE表示成功；FALSE表示失败。获取错误码调用 NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_SENDFAIL	0x20000006	发送数据失败
NET_WAITTIMEOUT	0x20000002	接收配置数据超时
NET_NOCONNECTED	0x2000000d	与IPC连接断开
NET_XMLCREATEORPARSEFAIL	0x2000000e	XML合成错误

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.7 设备维护

2.4.7.1 NETSDK_ReBoot

【函数原型】BOOL NETSDK_ReBoot(HANDLE hIPC)

【功能】远程重启设备。

【参数】

hIPC [in] IPC 登陆句柄，NETSDK_LoginIPC 返回值

【返回值】TRUE表示成功；FALSE表示失败。获取错误码调用 NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_SENDFAIL	0x20000006	发送数据失败
NET_WAITTIMEOUT	0x20000002	接收配置数据超时
NET_NOCONNECTED	0x2000000d	与IPC连接断开

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.7.2 NETSDK_Restore

【函数原型】BOOL NETSDK_Restore(HANDLE hIPC)

【功能】恢复设备默认设置。

【参数】

hIPC [in] IPC 登陆句柄，NETSDK_LoginIPC 返回值

【返回值】TRUE表示成功；FALSE表示失败。获取错误码调用 NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_SENDFAIL	0x20000006	发送数据失败
NET_WAITTIMEOUT	0x20000002	接收配置数据超时
NET_NOCONNECTED	0x2000000d	与IPC连接断开

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.7.3 NETSDK_Upgrade

*【函数原型】HANDLE NETSDK_Upgrade(HANDLE hIPC, const char *szFilePath, DEFUPGRADEPOC pUpgrqadeProc, void *param)*

【功能】升级 IPC 软件。

【参数】

hIPC [in] IPC 登陆句柄，NETSDK_LoginIPC 返回值

szFilePath [in] 本地升级软件文件路径

pUpgrqadeProc [in] 升级回调函数，可在此函数中获取升级进度及升级状态

param [in] 升级回调函数用户数据

【回调函数】

*typedef int (__stdcall *DEFUPGRADEPOC)(int xfered, void *arg);*

【回调函数参数】

xfered [out] 已上传的文件字节数

arg [out] 用户数据

【返回值】升级句柄，非NULL表示成功；NULL表示失败。升级完成后，需调用NETSDK_CloseUpgradeHandle函数来关闭句柄。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_INVALIDFILE	0x20000017	升级文件打开失败
NET_CREATTHREADFAIL	0x20000009	创建线程失败

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

函数成功返回后，可在回调函数中调用 NETSDK_GetUpgradeProgress 来获取升级文件上传进度，还可调用 NETSDK_GetUpgradeState 来获取升级状态，当状态为 UPGRADE_SUCCESS 时，表示升级成功，这是可调 NETSDK_CloseUpgradeHandle

函数关闭升级句柄。但是，不要在回调函数中直接调 NETSDK_CloseUpgradeHandle

函数，而是通知其它线程去关闭。不然会导致死锁。因为关闭函数要等待升级线程结束，而回调函数是在升级线程中被调用的。

2.4.7.4 NETSDK_UpgradeEx

【函数原型】HANDLE NETSDK_UpgradeEx(const char szIP, const char* szUser, const char* szPwd, const char *szFilePath, DEFUPGRADEPOC pUpgrqadeProc, void *param)*

【功能】升级 IPC 软件，与 NETSDK_Upgrade 函数不同的是，NETSDK_UpgradeEx 在未登陆情况下升级。

【参数】

szIP [in]要升级的 IPC 的 IP

szUser [in] 要升级的 IPC 的登陆用户名

szPwd [in] 要升级的 IPC 的登陆密码

szFilePath [in] 本地升级软件文件路径

pUpgrqadeProc [in] 升级回调函数，可在此函数中获取审计进度及升级状态

param [in] 升级回调函数用户数据

【回调函数】

*typedef int (__stdcall *DEFUPGRADEPOC)(int xfered, void *arg);*

【回调函数参数】

xfered [out] 已上传的文件字节数

arg　　[out] 用户数据

【返回值】升级句柄，非NULL表示成功；NULL表示失败。升级完成后，需调用NETSDK_CloseUpgradeHandle函数来关闭句柄。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_INVALIDFILE	0x20000017	升级文件打开失败
NET_CREATTHREADFAIL	0x20000009	创建线程失败

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

见 NETSDK_Upgrade 函数使用方法。

2.4.7.5　NETSDK_CloseUpgradeHandle

【函数原型】BOOL NETSDK_CloseUpgradeHandle(HANDLE hUpgrade)

【功能】关闭升级句柄，清理升级使用的资源。

【参数】

hUpgrade　　[in]要关闭的升级句柄，由 NETSDK_Upgrade 或 NETSDK_UpgradeEx 返回。

【返回值】TRUE表示成功，FALSE表示失败。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

【注意】不要在升级回调函数中直接调 NETSDK_CloseUpgradeHandle

函数，而应在回调函数中通知其它线程去关闭。不然会导致死锁。因为关闭函数要等待升级线程结束，而回调函数是在升级线程中被调用的。

2.4.7.6 NETSDK_GetUpgradeProgress

【函数原型】LONG NETSDK_GetUpgradeProgress(HANDLE hUpgrade)

【功能】获取升级或上传的文件上传进度。

【参数】

hUpgrade [in]升级或上传 logo 的句柄，由 NETSDK_Upgrade、NETSDK_UpgradeEx 返回。

【返回值】函数失败返回-1，成功则返回0~100的文件上传进度值。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.7.7 NETSDK_GetUpgradeState

【函数原型】LONG NETSDK_GetUpgradeState(HANDLE hUpgrade)

【功能】获取升级或上传文件的状态。

【参数】

hUpgrade [in]升级或上传文件的句柄，由 NETSDK_Upgrade、NETSDK_UpgradeEx 返回。

【返回值】升级或上传的状态。如下：

状态	状态值	状态信息
UPGRADE_IDLE	-1	无效值
UPGRADE_DO	0	准备连接FTP，上传文件

UPGRADE_SUCCESS	1	升级或上传logo成功结束
UPGRADE_UPLOADING	2	正在上传升级文件或logo
UPGRADE_ABORT	3	升级或上传被取消（通过调用NETSDK_CloseUpgradeHandle）
UPGRADE_FAILD	4	升级或上传失败
UPGRADE_FAILD_PRIVILEGE	5	由于用户权限不够，升级或上传失败
UPGRADE_CONNECTION_FAILD	6	FTP连接失败
UPGRADE_PROCESS	7	暂未使用
UPLOADE_FAILED_INCORRECTFORMAT	8	格式错误，升级或上传失败
UPGRADE_FAILED_FILENOTEXIST	9	文件不存在，升级或上传失败
UPGRADE_FAILED_OPENFILEERROR	10	读文件错误，升级或上传失败
UPGRADE_FAILED_READFILEHEADERERROR	11	文件头错误，升级或上传失败
UPGRADE_FAILED_INCORRECTFILETYPE	12	文件类型错误，升级或上传失败
UPGRADE_FAILED_CHECKFIRMWARECRCERROR	13	Firmware或CRC校验错误，升级失败
UPGRADE_FAILED_INCORRECTBOARDTYPE	14	文件板类型错误，升级失败
UPGRADE_FAILED_FIRMWAREISNOTNEW	15	文件Firmware版本错误，升级失败
UPGRADE_FAILED_TIMEOUT	16	等待超时，升级或上传失败

获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.7.8 NETSDK_CheckUpgradeFile

*【函数原型】LONG NETSDK_CheckUpgradeFile(const char *szFilePath)*

【功能】校验升级文件是否正确。

【参数】

szFilePath [in]文件路径。

【返回值】返回0表示校验成功，返回其它值表示校验失败。返回值列表如下：

宏定义	值	信息
	0	校验成功
	-1	文件路径无效或读文件失败，导致校验失败
UPGRADE_FAILED_READFILEHEADERERROR	11	读升级文件头错误
UPGRADE_FAILED_INCORRECTFILETYPE	12	校验魔术字失败
UPGRADE_FAILED_CHECKFIRMWARECRCERROR	13	校验CRC失败
UPGRADE_FAILED_FIRMWAREISNOTNEW	15	校验Firmware失败

如果返回值为-1，可调用NETSDK_GetLastError获取错误码。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_INVALIDFILE	0x20000017	读取文件失败

【声明头文件】IPCNetSDK_Interface.h

2.4.7.9 NETSDK_CloseUploadHandle

【函数原型】BOOL NETSDK_CloseUploadHandle(HANDLE hUpload)

【功能】关闭上传 Logo 返回的句柄，清理上传使用的资源。

【参数】

hUpload [in]要关闭的上传 Logo 返回的句柄，由 NETSDK_UploadLogo 返回。

【返回值】TRUE表示成功，FALSE表示失败。获取错误码调用 NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

【注意】不要在上传回调函数中直接调 NETSDK_CloseUploadHandle 函数，而应在回调函数中通知其它线程去关闭。不然会导致死锁。因为关闭函数要等待上传线程结束，而回调函数是指上传线程中被调用的。

2.4.7.10 NETSDK_StartSearchUnits

*【函数原型】HANDLE NETSDK_StartSearchUnits(DEFSEARCHUNITS pfSearchCallback, void *pUser, const char *szLocalIP);*

【功能】广播扫描局域网内的 IPC。

【参数】

pfSearchCallback [in]IPC 扫描回调函数，用于接收扫描到的 IPC 信息

pUser [in]回调函数用户数据

szLocalIP [in]本地 IP。可传入 NULL，此时将使用本计算机的 IP 来扫描

【回调函数】

```
typedef void (__stdcall *DEFSEARCHUNITS)(DeviceInfo_t *pDeviceInfo,

int iFlag,

void *pUser);
```

【回调函数参数】

pDeviceInfo [out] 扫描到的 IPC 信息，详见 DeviceInfo_t

iFlag [out] 扫描结束标志，详见 eSearchCallbackFlag

pUser [out] 用户数据

【返回值】返回NULL表示失败，否则表示成功。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_BROADCASTING_SHOULDWAIT	0x20000018	本进程有一个网络广播在进行，需稍后再试
NET_CREATTHREADFAIL	0x20000009	创建线程失败
NET_CREATESOCKETFAIL	0x20000008	创建套接字失败
NET_BINDFAIL	0x20000019	套接字绑定失败

【声明头文件】IPCNetSDK_Interface.h

【使用方法】调用本函数后，传递的回调函数中接收扫描到的 IPC，用回调函数的的 iFlag 判断扫描是否结束。扫描结束时，可调 NETSDK_StopSearchUnits

函数关闭扫描函数返回的句柄。但是，不要在回调函数中直接调 NETSDK_StopSearchUnits 函数，而应通知其它线程去关闭。不然会导致死锁。因为关闭函数要等待扫描线程结束，而回调函数是在扫描线程中被调用的。

2.4.7.11 NETSDK_StopSearchUnits

```
【函数原型】BOOL NETSDK_StopSearchUnits(HANDLE hSearch);
```

【功能】在 IPC 扫描结束或扫描中停止扫描。

【参数】

hSearch [in]IPC 扫描句柄，由函数返回 NETSDK_StartSearchUnits

【返回值】返回TRUE表示成功，FALSE表示失败。获取错误码调用 NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

【注意】不要在回调函数中直接调 NETSDK_StopSearchUnits 函数，而应通知其它线程去关闭。不然会导致死锁。因为关闭函数要等待扫描线程结束，而回调函数是在扫描线程中被调用的。

2.4.7.12 NETSDK_AbatchSetIpCfg

*【函数原型】BOOL NETSDK_AbatchSetIpCfg(char *szFirstIP, AbatchIPCfg_t *pAbatchIfaceParam, unsigned long deviceNnm, UserLoginInfo_t *pUserLoginInfo, char *szLocalIp);*

【功能】广播批量 IPC 网络设置，包括 IP、网关、子网掩码、DNS。函数可设置局域网内 IP 相同的 IPC 网络信息。

【参数】

szFirstIP [in]批量设置的 IP 首地址

pAbatchIfaceParam [in，out]指向网络批量设置信息指针，用于存储 deviceNnm 个 IPC 的网络信息，包括 Mac、网关、子网掩码、DNS，其中 Mac 是必须的，但无需包含 IP，函数会自动计算 IP。pAbatchIfaceParam 的 result 自动表设置的结果，函数返回后，可用此字段判断是否设置成功

deviceNnm [in]指示 pAbatchIfaceParam 包含信息量，即要设置的 IPC 数量

pUserLoginInfo [in]用户信息，要求包含用户名和密码，批量包含的所有 IPC 都使用此用户设置

szLocalIp [in]进行设置的本地计算机 IP。可设为 NULL，函数会默认使用本机 IP

【返回值】全部设置成功返回TRUE，否则返回FALSE。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_BROADCASTING_SHOULDWAIT	0x20000018	本进程有一个网络广播在进行，需稍后再试
NET_CREATESOCKETFAIL	0x20000008	创建套接字失败
NET_BINDFAIL	0x20000019	套接字绑定失败

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

【注意】本函数是同步设置，设置完所有 IPC 后返回。

2.4.7.13 NETSDK_SetIpCfg

*【函数原型】BOOL NETSDK_SetIpCfg(NetIPCfg_t *pIfaceParam, UserLoginInfo_t *pUserLoginInfo, char *szLocalIp);*

【功能】广播方式 IPC 网络设置，包括 IP、网关、子网掩码、DNS。函数可设置局域网内 IP 相同的 IPC 网络信息。

【参数】

pIfaceParam [in]指向网络设置信息指针，包括 IP、网关、子网掩码、DNS。还需包含要设置的 IPC 的 Mac

deviceNnm [in]指示 pAbatchIfaceParam 包含信息量，即要设置的 IPC 数量

pUserLoginInfo [in] 用户信息，要求包含用户名和密码

szLocalIp [in]进行设置的本地计算机 IP。可设为 NULL，函数会默认使用本机 IP

【返回值】设置成功返回TRUE，否则返回FALSE。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_BROADCASTING_SHOULDWAIT	0x20000018	本进程有一个网络广播在进行，需稍后再试
NET_CREATESOCKETFAIL	0x20000008	创建套接字失败
NET_BINDFAIL	0x20000019	套接字绑定失败

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

【注意】本函数是同步设置，设置完成后返回。

2.4.7.14 NETSDK_GetIpCfg

*【函数原型】BOOL NETSDK_GetIpCfg(NetIPCfg_t *pIfaceParam, char *szLocalIp);*

【功能】广播获取 IPC 网络设置，包括 IP、网购、子网掩码、DNS。函数可获取局域网内 IP 相同的 IPC 网络信息。

【参数】

pIfaceParam [in，out]指向获取的网络信息指针，包括 IP、网关、子网掩码、DNS。作为传入值，需包含要获取的 IPC 的 Mac

szLocalIp [in]进行设置的本地计算机 IP。可设为 NULL，函数会默认使用本机 IP

【返回值】全部设置成功返回TRUE，否则返回FALSE。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息

NET_INVALIDPARAMETER	0x20000003	参数错误
NET_BROADCASTING_SHOULDWAIT	0x20000018	本进程有一个网络广播在进行，需稍后再试
NET_CREATESOCKETFAIL	0x20000008	创建套接字失败
NET_BINDFAIL	0x20000019	套接字绑定失败

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

【注意】本函数是同步获取，获取完成后返回。

2.4.8 语音对讲

2.4.8.1 NETSDK_StartVoiceCom

*【函数原型】HANDLE NETSDK_StartVoiceCom(HANDLE hIPC, DEFVOICEDATAPROC pVoiceDataProc, void *param);*

【功能】打开双向语音对讲。

【参数】

hIPC [in]IPC 登陆句柄，NETSDK_LoginIPC 的返回值

pVoiceDataProc [in]回调函数，用于接收来自 IPC 和本地麦的音频数据

param [in] 用户信息

【返回值】返回NULL表示失败，否则返回对讲句柄。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_LOADPLAYDLLFAIL	0x2000000f	加载播放动态库失败
NET_OPENAUDIORECORDFAIL	0x20000015	开启获取本地麦音频失败

NET_CREATTHREADFAIL	0x20000009	创建线程失败
NET_NOCONNECTED	0x2000000d	与IPC连接断开
NET_SENDFAIL	0x20000006	发送数据失败
NET_RECVFAIL	0x20000007	接收数据失败
NET_WRONGDATA	0x2000000c	接收数据格式错误或与期望不符
NET_CREATESOCKETFAIL	0x20000008	创建套接字失败
NET_CONNECTFAIL	0x20000005	网络连接失败
NET_CREATEAUDIOPLAYERFAIL	0x20000015	开启音频播放失败

【声明头文件】IPCNetSDK_Interface.h

2.4.8.2 NETSDK_StopVoiceCom

【函数原型】BOOL NETSDK_StopSearchUnits(HANDLE hSearch);

【功能】关闭语音对讲。

【参数】

hSearch [in]语音对讲句柄，由 NETSDK_StartVoiceCom 返回

【返回值】成功返回TRUE，否则返回FALSE。获取错误码调用NETSDK_GetLastError。

错误类型	错误值	错误信息
NET_INVALIDPARAMETER	0x20000003	参数错误

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.9 获取错误信息

2.4.9.1 NETSDK_GetLastError

【函数原型】unsigned long NETSDK_GetLastError()

【功能】返回本线程最后操作的错误码。

【参数】

无

【返回值】本线程最后操作的错误码。错误码见下表：

错误	值	错误信息
NET_NOERROR	0	无错误
NET_UNINIT	0x20000001	SDK未初始化
NET_WAITTIMEOUT	0x20000002	等待接收数据超时
NET_INVALIDPARAMETER	0x20000003	参数错误
NET_MEMORYALLOCFAIL	0x20000004	分配内存失败
NET_CONNECTFAIL	0x20000005	网络连接失败
NET_SENDFAIL	0x20000006	发送数据失败
NET_RECVFAIL	0x20000007	接收数据失败
NET_CREATESOCKETFAIL	0x20000008	创建套接字失败
NET_CREATTHREADFAIL	0x20000009	试图创建线程失败
NET_CREATEVENTFAIL	0x2000000a	创建事件失败
NET_WRONGDATALEN	0x2000000b	接收数据的长度错误
NET_WRONGDATA	0x2000000c	接收数据格式错误或与期望不符
NET_NOCONNECTED	0x2000000d	与IPC连接断开

NET_XMLCREATEORPARSEFAIL	0x2000000e	XML合成错误
NET_LOADPLAYDLLFAIL	0x2000000f	加载播放动态库失败
NET_INITPLAYFAIL	0x20000010	播放初始化失败
NET_CAPTUREFAIL	0x20000011	播放解码库抓图失败
NET_UNINITPLAY	0x20000012	播放动态库未初始化。
NET_PLAYSOUNDFAIL	0x20000013	播放解码库打开声音失败
NET_PAUSESOUNDFAIL	0x20000014	播放解码库关闭声音失败
NET_OPENAUDIORECORDFAIL	0x20000015	开启获取本地麦音频失败
NET_CREATEAUDIOPLAYERFAIL	0x20000016	开启音频播放失败
NET_INVALIDFILE	0x20000017	文件打开失败
NET_BROADCASTING_SHOULDWAIT	0x20000018	本进程有一个网络广播在进行，需稍后再试
NET_BINDFAIL	0x20000019	套接字绑定失败
NET_IPC_BAD_HEADLINE	0x20002000	IPC反馈，请求的用户名或密码、序列号为空
NET_IPC_NOSUPPORT	0x20002001	IPC反馈，未知的请求
NET_IPC_NOCONTENT	0x20002002	IPC反馈，请求内容为空
NET_IPC_ERRORDATA	0x20002003	IPC反馈，数据错误
NET_IPC_INVALID_CMD	0x20002004	IPC反馈，无效的命令
NET_IPC_MAX_CONNECTION	0x20002005	IPC反馈，IPC连接已满
NET_IPC_PRIVILEGE_INVALID	0x20002006	IPC反馈，用户权限不足
NET_IPC_INVALID_USER	0x20002007	IPC反馈，无效的用户

NET_IPC_VIDIO_NOSOURCE	0x20000193	IPC反馈，IPC播放数已满
NET_IPC_VIDIO_SENSERERR	0x20000194	IPC反馈，IPC的Senser错误

【声明头文件】IPCNetSDK_Interface.h

2.4.10 SDK 调试

2.4.10.1 NETSDK_SetLog

*【函数原型】void NETSDK_SetLog(unsigned long ulLogWriteType, unsigned long ulContent,const char *szIPFilter, const char *szFilePath)*

【功能】控制 SDK 的 Log 输出，包括输出方式、输出内容。

【参数】

ulLogWriteType [in] Log 输出方式，包括文件输出和 TRACE 输出（可用 Dbgview 工具显示 TRACE 输出），0值表示不输出。可同时进行这两种输出。输出方式如下表：

宏定义	值	含义
LOG_WRITETYPE_TRACE	0x00000001	TRACE输出
LOG_WRITETYPE_FILE	0x00000002	文件输出

ulContent [in] Log 输出内容种类，0值表示不输出。几种内容可同时输出。输出内容见下表：

宏定义	值	含义
LOG_CONTENT_DEBUG	0x00000001	输出调试信息
LOG_CONTENT_ERROR	0x00000002	输出异常信息
LOG_CONTENT_RECEIVEBUF	0x00000004	输出网络接收到的信息
LOG_CONTENT_SENDBUF	0x00000008	输出网络发送的信息

szIPFilter　[in] 设备的 IP 地址，用于选择要需要输出的设备。如要选择多个设备，可用”,”隔开多个设备的 IP，最多可包含5个 IP 地址。 “all” 或 NULL 值表示不区分设备，全部输出。

szFilePath　[in] 文件输出的路径和文件名，采用文件输出时有效。

【返回值】无

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

2.4.11　PTZ 控制

【函数原型】bool NETSDK_PtzCtrl(HANDLE hIPC, const PtzCtrl_t ctrl)*

【功能】远程控制 ptz

【参数】

hIPC　*hIPC*　[in] IPC 登陆句柄，NETSDK_LoginIPC 返回值

PtzCtrl_t　[in]控制命令：

【返回值】【返回值】TRUE表示成功；FALSE表示失败。获取错误码调用 NETSDK_GetLastError

【声明头文件】IPCNetSDK_Interface.h

【使用方法】

3 编程导引

3.1.基本流程说明

每个模块的功能实现时初始化 SDK、注册设备用户、注销用户和释放 SDK 资源这 4 个流程是必不可少的。

初始化 SDK：对整个网络 SDK 系统的初始化，内存预分配等操作。

用户注册设备：实现用户的注册功能，注册成功后，返回的用户句柄作为其他功能操作的唯一标识。

3.2.预览流程

3.2.1. 预览

这些模块为预览提供辅助功能，它们之间相互独立，各自完成相应的功能。

- 声音控制功能主要实现独占声音的开启、暂停。相关接口是：NETSDK_PlaySound、NETSDK_PauseSound 等。
- 实时流数据捕获主要实现数据回调的功能，可供用户后续处理。
- 录像模块与抓图模块主要实现图像捕获的功能，图像可以存储为 BMP。相关接口是： NETSDK_CaptureBMP 等。
- 云台控制模块主要是在开启预览的前提下实现对云台控制的操作功能。

3.2.2. 实时流解码模式

直接播放模式

在预览接口中预览参数的播放窗口句柄赋成有效句柄，则由 SDK 实现解码功能。在初始化 SDK 和注册设备两步骤后，直接调用启动预览和停止预览接口。

【示例代码】

```
#include <stdio.h>

#include <string.h>

#include <Windows.h>

#include " IPCNetSDK_Interface.h "

void _stdcall NetSDKMessage (HANDLE hIPC, DWORD dwMsg, void* lpData, int nLength, void* param)

{
```

```
        switch (dwMsg)
        {
        case INFOTYPE_ALARM:
            LPALARM_INFO pAlarmInfo = (LPALARM_INFO)lpData;
            switch (pAlarmInfo->lCommand)
{
case COMMAND_ALARM_COMMON:
{
PALARMINFO_COMMON pInfo = (PALARMINFO_COMMON)pAlarmInfo->pAlarmInfo;
//处理上报告警
break;
}
}
            break;
        case INFOTYEP_EXCEPTION:
            PEXCEPTION_INFO pExceptionInfo = (PEXCEPTION_INFO)lpData;
        if (pExceptionInfo->uExceptionType == EXCEPTION_EXCHANGE)
        {
        //发生链接断开异常
        }
else if (pExceptionInfo->uExceptionType == EXCEPTION_RECONNECT)
        {
```

```
        //异常断开重连成功
        }
            break;
        default:
            break;
        }
}
void main()
{
        NETSDK_Init();
char *userIPAddr = “192.168.27.215”;
char *szUserName = “admin”;
char *szPassword = “admin”;
        int port = 1115;
HANDLE hIPC = NETSDK_LoginIPC(userIPAddr, , szUserName, szPassword, NULL, NULL);
        if (hIPC == NULL)
        {
         printf("Login to Device failed! Error : %d\n",
           NETSDK_GetLastError());
return;
        }
       HWND hWnd = GetConsoleWindow();
```

```
 OPEN_CHANNEL_INFO playInfo;

      memset(playInfo, 0, sizeof(OPEN_CHANNEL_INFO));

   playInfo.nChannel = STREAM_IDX_VIDEO_MAIN;

   playInfo.lpParam = NULL;

   playInfo.lpProc = NULL;

   playInfo.hPlayWnd = hWnd;

HANDLE lPlayHandle = NETSDK_OpenChannel(theApp.m_lLoginID, &playInfo);

      if (lPlayHandle == NULL)

      {

        printf ("Start Play failed! Error : %d\n", NETSDK_GetLastError());

        NETSDK_LogoutIPC (hIPC);

      NETSDK_Cleanup ();

        return;

      }

      Sleep(10000);

      NET_StopRealplay(lPlayHandle);

      NETSDK_LogoutIPC (hIPC);

      NETSDK_Cleanup ();

      return;

}
```

回调播放模式

用户可以通过设置预览接口中预览参数的播放窗口句柄为空值，并通过调用捕获数据的接口（即设置接口中的回调函数），获取码流数据进行后续解码播放处理。

【示例代码】

```
#include <stdio.h>

#include <string.h>

#include <Windows.h>

#include " IPCNetSDK_Interface.h "

#include “tfipcplayer_interface.h”

HANDLE hplayer;

HWND hWnd;

void _stdcall RealVideoDataCallBack(HANDLE hChannel, BYTE* lpData, DWORD nLength, BYTE *extra,
DWORD extrelen, int nFrameType,void* param)

{

if(hChannel == NULL)

{

   return;

}

   switch (nFrameType)

   {

   case NET_MEDIAINFO_DATA:

      {

         if (extra == NULL || extrelen <= 0)
```

```
        {
            return;
        }
        hplayer  = IPCPLAYER_CreatePlayer(hWnd, extra, extrelen);
    }
    break;
  case NET_VIDEO_DATA:
    {
        NET_FRAME_HEAD *framehead = (NET_FRAME_HEAD *)extra;
        if (lpData == NULL || nLength <= 0 || extra == NULL ||
extrelen <= 0)
        {
            return;
        }
     IPCPLAYER_InputVideoData(hplayer, lpData, nLength, framehead.ulTimeStamp,
framehead.usFrameType, 0);
    }
    break;
  case NET_AUDIO_DATA:
    {
        if (pBuffer == NULL || dwBufSize <= 0)
        {
```

```
            return;
          }
IPCPLAYER_InputAudioData(hplayer, lpData, nLength);
        }
       }
       break;
    }
}
void main()
{
       NETSDK_Init();
char *userIPAddr = “192.168.27.215”;
char *szUserName = “admin”;
char *szPassword = “admin”;
       int port = 1115;
HANDLE hIPC = NETSDK_LoginIPC(userIPAddr, , szUserName, szPassword, NULL, NULL);
       if (hIPC == NULL)
       {
        printf("Login to Device failed! Error : %d\n",
          NETSDK_GetLastError());
return;
       }
```

```
OPEN_CHANNEL_INFO playInfo;

        memset(playInfo, 0, sizeof(OPEN_CHANNEL_INFO));

    playInfo.nChannel = STREAM_IDX_VIDEO_MAIN;

    playInfo.lpParam = NULL;

    playInfo.lpProc = RealDataCB;

    playInfo.hPlayWnd = NULL;

HANDLE lPlayHandle = NETSDK_OpenChannel(theApp.m_lLoginID, &playInfo);

        if (lPlayHandle == NULL)

        {

          printf ("Start Play failed! Error : %d\n", NETSDK_GetLastError());

          NETSDK_LogoutIPC (hIPC);

        NETSDK_Cleanup ();

          return;

        }

        Sleep(10000);

        NET_StopRealplay(lPlayHandle);

        NETSDK_LogoutIPC (hIPC);

        NETSDK_Cleanup ();

        return;

}
```

3.3.参数配置流程

实现参数配置首先必须做好初始化 SDK 和用户注册这两个步骤，将用户注册接口返回的句柄作为配置接口的首个参数。建议在每次设置某类参数之前，先调用获取参数的接口得到完整的参数结构，修改需要更改的参数，作为设置参数接口中的输入参数，最后调用设置参数接口，返回成功即设置成功。

【示例代码】

```
#include <stdio.h>

#include <string.h>

#include <Windows.h>

#include " IPCNetSDK_Interface.h "

int main()

{

        NETSDK_Init();

char *userIPAddr = “192.168.27.215”;

char *szUserName = “admin”;

char *szPassword = “admin”;

        int port = 1115;

HANDLE hIPC = NETSDK_LoginIPC(userIPAddr, , szUserName, szPassword, NULL, NULL);

        if (hIPC == NULL)

        {

         printf("Login to Device failed! Error : %d\n",

           NETSDK_GetLastError());
```

```
return;
        }
    TimeInfo_t timeinfo;
    ZeroMemory(&timeinfo, sizeof(timeinfo));
    timeinfo.Year = 2013 - 2000;
    timeinfo.Month = 12;
    timeinfo.Day = 12;
    timeinfo.Hour = 12;
    timeinfo.Minute = 12;
    timeinfo.Second = 12;
BOOL bRet = FALSE;
bRet  =  NETSDK_SetParam (hIPC, CMD_SET_TIME, 0, 0, 0, meinfo, sizeof(timeinfo));
    if (bRet == FALSE)
    {
        printf("SetConfig failed! Error : %d\n",
            NETSDK_GetLastError());
            NETSDK_LogoutIPC (hIPC);
        NETSDK_Cleanup ();
return -1;
    }
int iLen = sizeof(meinfo);
bRet = NETSDK_GetParam(hIPC, CMD_SET_TIME, 0, 0, 0, meinfo, &iLen);
```

```
    if (bRet == FALSE)
    {
        printf("GetConfig failed! Error : %d\n",
            NETSDK_GetLastError());
    }
        NET_Logout(lUserID);
        NET_Cleanup();
        return 0;
}
```

3.4.报警流程

登陆时，SDK 会主动连接设备，并向设备订阅报警，设备发生报警会立即发送给 SDK。示例代码可参见“浏览流程”的 “直接播放模式”中的示例代码。

3.5.单向语音

单向语音是指从客户端通过 SDK 向 IPC 发送语音数据。

客户端语音采样参数：

Format = PCM;

nChannels = 1;
nSamplesPerSec = 8000;
wBitsPerSample = 16;

在 window 环境下，IPCPLAYER_OpenAudioRecord 函数的回调音频数据可直接通过 NETSDK_InputAudioData 接口发送给 IPC。

接口：

NET_EXPROT HANDLE NETSDK_StartAudio(HANDLE hIPC)//创建语音通道

NET_EXPROT void NETSDK_InputAudioData(HANDLE hAudio,char* lpData,int nLength)//发送音频数据

NET_EXPROT void NETSDK_StopAuido(HANDLE hAudio)//销毁语音通道

调用流程：

NETSDK_LoginIPC

NETSDK_StartAudio

NETSDK_InputAudioData

NETSDK_StopAuido

NETSDK_LogoutIPC

3.6. Ptz 控制

用户可以远程控制 IPC 的 ptz 功能， 包括变倍、聚焦、光圈、雨刷、预置位等。

Ptz 控制流程参考如下：

NETSDK_LoginIPC

NETSDK_GetParam //cmd 为 CMD_GET_PTZCFG, 获取当前 ptz 获取

NETSDK_SetParam //cmd 为 CMD_SET_PTZCFG 设置 ptz 参数

NETSDK_PtzCtrl

NETSDK_LogoutIPC