ipcam sdk param dev guide

(v.1.0.1)

[一、 修订日志............................................................................................................................ 3](#_Toc26499)

[二、 约订.................................................................................................................................... 4](#_Toc1822)

[三、 功能列表............................................................................................................................ 5](#_Toc28744)

[四、 接口说明............................................................................................................................ 9](#_Toc2252)

[五、 结构体说明.......................................................................................................................1](#_Toc26235)1

六、 回调...................................................................................................................................29

1. 参数...................................................................................................................................31
2. 宏定义...............................................................................................................................32

# 修订日志

|  |  |  |  |
| --- | --- | --- | --- |
| V1.0.0 | 2017/08/25 | 开始 | Vincent.Yeh |
| v.1.0.1 | 2017/08/26 | 添加WIFI热点搜索 ipcam\_wifi\_ap\_search说明 | 叶 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 约订

1. 一般情况下获取接口：ipcam\_get\_device\_config，设置接口：ipcam\_set\_device\_config，如果输入输出都涉及到参数则使用ipcam\_device\_config。
2. 一般数组属性都有一个\_\_pri\_xxxxxCount的属性，用来限定数组xxxxx的有效个数
3. 有些情况还需调用接口注册回调，并在回调中做出相应的处理

# 功能列表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 枚举 | 协议结构 | 说明 |
| 用户管理 | IPCNET\_GET\_USER\_ALL\_INFO\_REQ | [IPCNetGetUserGroupCfg\_st](#_IPCNetGetUserGroupCfg_st) | 使用接口：  （获取所有用户信息）  ipcam\_get\_device\_config |
| IPCNET\_GET\_USER\_INFO\_REQ | 输入：[IPCNetUserInfoReq\_st](#_IPCNetUserInfoReq_st)  输出：[IPCNetUserInfo\_st](#_IPCNetUserInfo_st) | 使用接口：  （获取用户信息）  ipcam\_device\_config |
| IPCNET\_SET\_USER\_INFO\_REQ | [IPCNetUserInfo\_st](#_IPCNetUserInfo_st) | 使用接口：  （设置用户信息）  ipcam\_set\_device\_config |
| 有线网络 | IPCNET\_ETH\_GET\_REQ | [IPCNetEthConfig\_st](#_IPCNetEthConfig_st) | 使用接口：  （获取有线网络配置）  ipcam\_get\_device\_config |
| IPCNET\_ETH\_SET\_REQ | [IPCNetEthConfig\_st](#_IPCNetEthConfig_st) | 使用接口：  （设置有线网络配置）  ipcam\_set\_device\_config |
| 移动网线 | IPCNET\_NETWORK\_MOBILE\_GET\_REQ | [Ipcnet3GInfo\_st](#_Ipcnet3GInfo_st) | 使用接口：  （获取移动网络配置）  ipcam\_get\_device\_config |
| IPCNET\_NETWORK\_MOBILE\_SET\_REQ | [Ipcnet3GInfo\_st](#_Ipcnet3GInfo_st) | 使用接口：  （设置移动网络配置）  ipcam\_set\_device\_config |
| 无线网络 | IPCNET\_NETWORK\_WIFI\_GET\_REQ | [IPCNetWirelessConfig\_st](#_IPCNetWirelessConfig_st) | 使用接口：  （获取设备当前WIFI信息）  ipcam\_get\_device\_config |
| IPCNET\_NETWORK\_WIFI\_SET\_REQ | [IPCNetWirelessConfig\_st](#_IPCNetWirelessConfig_st) | 使用接口：  （配置设备WIFI信息）  ipcam\_set\_device\_config |
| WIFI热点搜索 | IPCNET\_NETWORK\_WIFI\_SEARCH\_GET\_REQ | [IpcnetNetworkWirelessSearch\_st](#_IpcnetNetworkWirelessSearch_st_1) | （WIFI搜索）  详见[ipcam\_wifi\_ap\_search](#_WIFI热点搜索 ipcam_wifi_ap_search) |
| 当前网络状态 | IPCNET\_GET\_NETWORKS\_STATUS\_REQ |  | 暂未实现 |
| 视频参数 | IPCNET\_VIDEO\_ENC\_GET\_REQ | [IPCNetVideoEncodeCfg\_st](#_IPCNetVideoEncodeCfg_st) | 使用接口：  （获取视频参数）  ipcam\_get\_device\_config |
| IPCNET\_VIDEO\_ENC\_SET\_REQ | [IPCNetVideoEncodeCfg\_st](#_IPCNetVideoEncodeCfg_st) | 使用接口：  （设置视频参数）  ipcam\_set\_device\_config |
| 音频参数 | IPCNET\_AUDIO\_ENC\_GET\_REQ | [IpcnetAudioEncodeCfg\_st](#_IpcnetAudioEncodeCfg_st) | 使用接口：  （获取音频参数）  ipcam\_get\_device\_config |
| IPCNET\_AUDIO\_ENC\_SET\_REQ | [IpcnetAudioEncodeCfg\_st](#_IpcnetAudioEncodeCfg_st) | 使用接口：  （设置音频参数）  ipcam\_set\_device\_config |
| 移动侦测 | IPCNET\_MOVE\_ALARM\_GET\_REQ | [IPCNETMoveAlarmCfg\_st](#_IPCNETMoveAlarmCfg_st) | 使用接口：  （获取移动侦测参数）  ipcam\_get\_device\_config |
| IPCNET\_MOVE\_ALARM\_SET\_REQ | [IPCNETMoveAlarmCfg\_st](#_IPCNETMoveAlarmCfg_st) | 使用接口：  （设置移动侦测参数）  ipcam\_set\_device\_config |
| 时间参数 | IPCNET\_GET\_TIME\_REQ | [IPCNetTimeCfg\_st](#_IPCNetTimeCfg_st) | 使用接口：  （获取时间参数）  ipcam\_get\_device\_config |
| IPCNET\_SET\_TIME\_REQ | [IPCNetTimeCfg\_st](#_IPCNetTimeCfg_st) | 使用接口：  （设置时间参数）  ipcam\_set\_device\_config |
| 报警上报 | 使用[ipcam\_regist\_alarm\_event\_callback](#_.报警上报 ipcam_regist_alarm_event_callback)注册上报回调 | [IPCNetAlarmMsgReport\_st](#_IPCNetAlarmMsgReport_st) | （报警上报）  IPCNET\_ALARM\_IO\_INPUT报警时：idx-IO序号，val-IO口的值 |
| 录像设置 | IPCNET\_AV\_RECO\_CONF\_GET\_REQ | 输入：[IPCNetRecordGetCfg\_st](#_IPCNetRecordGetCfg_st)  输出：[IPCNetRecordCfg\_st](#_IPCNetRecordCfg_st) | 使用接口：  （获取录像机设置）  ipcam\_device\_config |
| IPCNET\_AV\_RECO\_CONF\_SET\_REQ | [IPCNetRecordCfg\_st](#_IPCNetRecordCfg_st) | 使用接口：  （设置录像机参数）  ipcam\_set\_device\_config |
| PTZ设置 | IPCNET\_PTZ\_SET\_REQ | [IPCPtzCtrlMsg\_st](#_IPCPtzCtrlMsg_st) | 使用接口：  （设置PTZ参数）  ipcam\_set\_device\_config |
| 设备参数 | IPCNET\_GET\_SYS\_INFO | [IPCNetCamInfo\_st](#_IPCNetCamInfo_st) | 使用接口：  （获取设备参数）  ipcam\_get\_device\_config |
| IPCNET\_SET\_DEV\_INFO\_REQ | [IPCNetDevName\_st](#_IPCNetDevName_st) | 使用接口：  （获取设备名）  ipcam\_set\_device\_config |
| 磁盘参数 | IPCNET\_GET\_DISK\_CFG\_REQ | [IPCNetDiskInfo\_st](#_IPCNetDiskInfo_st) | 使用接口：  （获取磁盘参数）  ipcam\_get\_device\_config |
| 格式化磁盘 | IPCNET\_FORMAT\_EXFAT\_REQ | [IPCNetRet\_st](#_IPCNetRet_st) | （SD卡格式化）  详见[ipcam\_storage\_format](#_SD卡格式化 ipcam_storage_format) |
| 图像参数 | IPCNET\_GET\_CAM\_PIC\_CFG\_REQ | [IPCNetCamColorCfg\_st](#_IPCNetCamColorCfg_st) | 使用接口：  （获取图像配置）  ipcam\_get\_device\_config |
| 强制I帧 | IPCNET\_VIDEO\_IFRAME\_REQ | [IPCNetVideoIFrame\_st](#_IPCNetVideoIFrame_st) | 使用接口：  （强制获取I帧）  ipcam\_set\_device\_config |
| 获取OSD | IPCNET\_GET\_OSD\_REQ | 输入：[IPCNetGetOsdCfg\_st](#_IPCNetGetOsdCfg_st)  输出：[IPCNetOsdCfg\_st](#_IPCNetOsdCfg_st) | 使用接口：  （获取OSD参数）  ipcam\_device\_config |
| 设置OSD | IPCNET\_SET\_OSD\_REQ | [IPCNetOsdCfg\_st](#_IPCNetOsdCfg_st) | 使用接口：  （设置OSD参数）  ipcam\_set\_device\_config |
| 图像翻转设置 | IPCNET\_GET\_OVERTURN\_REQ | [IPCNetPicOverTurn\_st](#_IPCNetPicOverTurn_st) | 使用接口：  （获取当前图像翻转状态）  ipcam\_get\_device\_config |
| IPCNET\_SET\_OVERTURN\_REQ | [IPCNetPicOverTurn\_st](#_IPCNetPicOverTurn_st) | 使用接口：  （设置图像翻转状态）  ipcam\_set\_device\_config |
| 预置点信息 | IPCNET\_GET\_PREPOINT\_REQ | [IPCNETPrePointList\_st](#_IPCNETPrePointList_st) | 使用接口：  （获取预支点信息）  ipcam\_get\_device\_config |
| IPCNET\_SET\_PREPOINT\_REQ | [IPCNETPointInfo\_st](#_IPCNETPointInfo_st) | 使用接口：  （设置预置点）  ipcam\_set\_device\_config |
| IPCNET\_OPERATE\_PREPOINT\_REQ | [IPCNETPointOpr\_st](#_IPCNETPointOpr_st) | 使用接口：  （对预置点进行操作）  ipcam\_set\_device\_config |
| FTP参数 | IPCNET\_GET\_FTP\_CFG\_REQ | [IPCNetFtpCfg\_st](#_IPCNetFtpCfg_st) | 使用接口：  （获取FTP参数）  ipcam\_get\_device\_config |
| IPCNET\_SET\_FTP\_CFG\_REQ | [IPCNetFtpCfg\_st](#_IPCNetFtpCfg_st) | 使用接口：  （设置FTP参数）  ipcam\_set\_device\_config |
| 设备设置 | IPCNET\_SET\_REBOOT\_REQ | NULL | 使用接口：  （设备重启）  ipcam\_set\_device\_config |
| IPCNET\_SET\_DEFAULT\_REQ | NULL | 使用接口：  （恢复出厂设置）  ipcam\_set\_device\_config |
| 设备对讲 | IPCNET\_TALK\_REQ | [IPCNetTalkReq\_t](#_IPCNetTalkReq_t) | 详见：[.设备对讲](#_.设备对讲) |
| 升级信息 | IPCNET\_UPGRADE\_REQ | [IPCNetUpgradeInfo\_st](#_IPCNetUpgradeInfo_st) | 使用接口：  （设置升级路径）  ipcam\_set\_device\_config |
| IPCNET\_GET\_UPGRADE\_URL\_REQ | [IPCNetUpgradeCfg\_st](#_IPCNetUpgradeCfg_st) | 使用接口：  （获取当前版本）  （注意：需用URL解密函数去解密LocalUrl和UpgradeUrl）  ipcam\_get\_device\_config |
| 录像回放 | IPCNET\_AV\_RECO\_LIST\_GET\_REQ | 输入：[IPCNetAvRecordInfoReq\_st](#_IPCNetAvRecordInfoReq_st)  输出：[IPCNetAvRecordInfoResp\_st](#_IPCNetAvRecordInfoResp_st) | 使用接口：  （获取录像回放列表）  ipcam\_get\_device\_config |
| IPCNET\_AV\_RECO\_LIST\_PAGE\_GET\_REQ | 输入：[IPCNetAvRecListPageReq\_t](#_IPCNetAvRecListPageReq_t)  输出：[IPCNetAvRecListPageResp\_t](#_IPCNetAvRecListPageResp_t) |
| IPCNET\_AV\_RECO\_OP\_REQ |  | 详见：[.录像回放](#_.录像回放) |
| 抓拍事件 |  |  |  |
| 人脸识别接口 | IPCNET\_FR\_GET\_USER\_REQ | 输出：IPCNetFrGetUserInfo\_t | 获取当前已经设置的用户信息，使用接口：ipcam\_get\_device\_config |
| IPCNET\_FR\_COLLECTION\_REQ | 输入：IPCNetFrCollectionReq\_st | 采集人脸信息，抓取的结果在回调中返回，type=PIC\_FR\_COLLECT，使用接口：ipcam\_get\_device\_config |
| IPCNET\_FR\_ADD\_USER\_BY\_COLLECT\_REQ | 输入：IPCNetFrAddCollectUser\_st | 根据采集的人脸信息添加用户，使用接口：ipcam\_get\_device\_config |
| IPCNET\_FR\_ADD\_USER\_BY\_VISIT\_HISTORY\_REQ | 输入：IPCNetFrImgName\_st | 从来访记录中添加用户 |
| IPCNET\_FR\_MODIFY\_USER\_INFO\_REQ | 输入：IPCNetFrModifyUserInfo\_st | 修改已保存的用户信息 |
| IPCNET\_FR\_DEL\_USER\_INFO\_REQ | 输入：IPCNetDelUserInfo\_st | 删除已保存的用户信息 |
| IPCNET\_FR\_GET\_VISIT\_HISTORY\_REQ | 输入：IPCNETGetVisitHistoryReq\_st | 获取来访者的历史记录，结果在回调中返回type=PIC\_FR\_VISIT\_HISTORY |
| IPCNET\_FR\_FUNC\_ENABLE\_REQ | 输入：IPCNETNetFrFuncEnabe\_st | 人脸识别功能开关 |
| ipcam\_regist\_fr\_event\_callback |  | 注册人脸识别的回调函数，详细见人脸识别回调部分 |

# 接口说明

## .WIFI热点搜索 ipcam\_wifi\_ap\_search

int ipcam\_wifi\_ap\_search(

[SEARCH\_WIFI\_AP\_CB](#_WIFI热点搜索回调：SEARCH_WIFI_AP_CB) cb,

void \*user\_data,

size\_t data\_len);

参数：

cb: 搜索热点出结果后会调用此回调

user\_data：方便用户在回调cb同时回调的用户数据, sdk会自行管理一份user\_data的拷贝。

data\_len：用户数据大小

说明:

Wifi搜索接口是异步的，调用搜索接口，当获得有搜索结果后回调会列出对应的热点信息，并在回调中处理数据

## .SD卡格式化 ipcam\_storage\_format

int ipcam\_storage\_format(

[STORAGE\_FORMAT\_CB](#_SD卡格式化回调) cb,

void \*user\_data,

size\_t data\_len);

参数：

cb: 格式化请求会调用此回调

user\_data: 方便用户在回调cb同时回调的用户数据

data\_len：用户数据大小

说明：

当外部有数据传进来会在回调中处理数据

## .报警上报 ipcam\_regist\_alarm\_event\_callback

int ipcam\_regist\_alarm\_event\_callback(

[IPCAM\_ALARM\_EVENT\_CB](#_报警上报回调) callback);

参数：

callback：当有报警时间传来会调用此回调

说明：

此函数只需在p2p初始化时注册回调函数即可，当报警事件来时在回调中处理数据

## .设备对讲

接口一:STAR SPEAK

HANDLE\_t ipcam\_adec\_start(

void \*reserve);

参数:

reserve：保留数据，一般情况为NULL

返回值：

返回一个句柄，若返回值小于0则开启失败

接口二:SEND SPEAK DATA

int ipcam\_send\_adec\_data(

HANDLE\_t handle,

char \*data,

size\_t data\_size,

void \*reserve);

参数:

handle：句柄

data：要发送的数据

data\_size：为数据长度

reserve：保留数据，一般情况为NULL

说明：

data：不包括头步数据，需要跳过头部数据发送

data\_size：则要减去头部数据长度

接口三:END SPEAK

int ipcam\_adec\_end(

HANDLE\_t handle);

参数：

handle：句柄

说明：

当获取到数据的时候调用接口一

发送数据时调用接口二

数据发送完成之后调用接口三

（接口一和接口三要成对出现）

## .录像回放

接口一：

int ipcam\_recorder\_play(

[IPCNET\_RECORD\_REQ\_t](#_IPCNET_RECORD_REQ_t) \*req);

参数：

req：播放请求

返回值：

代表着一个文件的id号

接口二：

int ipcam\_recorder\_ctrl(

int id,

int ctrl);

参数：

id：文件id号

ctrl：0: (恢复)播放 1: 暂停 2:结束

接口三：

int ipcam\_recorder\_speed(

int id,

int speed);

参数：

id：文件id号

speed：0-4，每增加1快进1/4

接口四：

int ipcam\_recorder\_stop(

int id);

参数：

id：文件id号

接口五：

int get\_ipcam\_recorder\_stream(

int id,

[STREAM\_FRAME\_INFO\_t](#_STREAM_FRAME_INFO_t) \*frameinfo,

char \*buffer,

unsigned int bufflen,

int \*getframeIndex);

参数：

id：文件id号

frameinfo：帧信息

buffer：数据流

bufflen：数据大小

getframeIndex：获取帧的索引

说明：

接口一：发送视频请求

接口二：发送命令到相对应的文件

接口三：控制文件播发

接口四：停止文件播放

## .抓拍事件

int ipcam\_snap\_shoot(

[IPCAM\_SNAP\_PIC\_CB](#_抓拍时间回调) cb,

void \*user\_data,

int data\_len);

参数：

cb：有抓拍时间后会调用此接口

user\_data：方便用户在回调cb同时回调的用户数据

data\_len：用户数据大小

说明：

# 结构体说明

## STREAM\_FRAME\_INFO\_t

typedef struct

{

uint16\_t vench;

uint16\_t payload;

uint16\_t frame\_type;

uint16\_t frame\_no;

uint32\_t frame\_size;

uint32\_t flag:1;

uint32\_t reserve:31;

uint64\_t timestamp;

}STREAM\_FRAME\_INFO\_t;

参数：

vench：视频编码通道

payload：

frame\_type：帧类型

frame\_no：帧序号

frame\_size：帧长

flag：目前录像回放用到这个flag，flag=1为录像文件全部回放完了

reserve：

timestamp：精确到微秒级时间戳

## IPCNET\_RECORD\_COMMAND\_e

typedef enum

{

IPCNET\_CMD\_RECORD\_PLAY,

IPCNET\_CMD\_RECORD\_STOP,

IPCNET\_CMD\_RECORD\_PAUSE,

IPCNET\_CMD\_RECORD\_REPLAY,

IPCNET\_CMD\_RECORD\_SPEED,

IPCNET\_CMD\_RECORD\_DRAG\_DROP,

}IPCNET\_RECORD\_COMMAND\_e;

## IPCNET\_RECORD\_REQ\_t

typedef struct

{

int ViCh;

int Command;

int recorder\_id;

char Filename[128];

[IPCNetDate\_st](#_IPCNetDate_st) Date;

[IPCNetTime\_st](#_IPCNetTime_st_1) Time;

int Args[2];

int \_\_pri\_ArgsCount;

}IPCNET\_RECORD\_REQ\_t;

参数：

ViCh：

Command：对应[IPCNET\_RECORD\_COMMAND\_e](#_IPCNET_RECORD_COMMAND_e)

recorder\_id：server端的id，IPCNET\_CMD\_RECORD\_STOP才用到

Filename：播放的文件名

Date：日期

Time：时间

Args：具体内容由Command决定

IPCNET\_CMD\_RECORD\_DRAG\_DROP:

Arg[0]为播放的起始时间(相对第一帧多少秒)，可省略，默认为开始 处播

IPCNET\_CMD\_RECORD\_SPEED:

Arg[0]为播放速度，可省略，默认为常速(1)

\_\_pri\_ArgsCount：

## IPCNetGetUserGroupCfg\_st

typedef struct//获取当前所有用户信息

{

[IPCNetGetUserInfo\_st](#_IPCNetGetUserInfo_st) User[10];

int \_\_pri\_UserCount;

}IPCNetGetUserGroupCfg\_st;

参数：

User：当前所有用户的信息

\_\_pri\_UserCount：当前所有用户个数

## IPCNetGetUserInfo\_st

typedef struct//获取到的用户信息

{

char User[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN   32)];

char Passwd[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN   32)];

}IPCNetGetUserInfo\_st;

参数：

User：用户名

Passwd：用户密码

## IPCNetUserInfo\_st

typedef struct//用户信息

{

char User[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN   32)];

char Passwd[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN   32)];

}IPCNetUserInfo\_st;

参数：

User：用户名

Passwd：用户密码

## IPCNetUserInfoReq\_st

typedef struct//用户名

{

char User[IPCNET\_STRING\_LEN];

}IPCNetUserInfoReq\_st;

参数：

User：用户名

## IPCNETMoveAlarmCfg\_st

typedef struct //移动侦测

{

T\_S32 ViCh;

[IPCNetDefendPeriodInfo\_st](#_IPCNetDefendPeriodInfo_st) Week[8];

[IPCNetVideoMoveInfo\_st](#_IPCNetVideoMoveInfo_st) MoveInfo;

[IPCNetMotionCellInfo\_st](#_IPCNetMotionCellInfo_st) MoveCell;

[IPCNetAlarmLinkagePolicyInfo\_st](#_IPCNetAlarmLinkagePolicyInfo_st) PolicyInfo;

}IPCNETMoveAlarmCfg\_st;

参数：

ViCh：镜头通道，ipc默认为0

Week[]：Week[0]:everyday Week[1]-Week[7]:since Sunday to saturday

MoveInfo：这里的协议数据太大了，不建议使用了

MoveCell：改成使用这里的数据

PolicyInfo：报警信息

说明：

属性MoveInfo不使用了，不必管它，使用MoveCell里的数据

## IPCNetDefendPeriodInfo\_st

typedef struct //布防时间段

{

T\_S32 Flag;

[IPCNetTimePeriod\_st](#_IPCNetTimePeriod_st) TimePeriod[2];

T\_S32 \_\_pri\_TimePeriodCount;

}IPCNetDefendPeriodInfo\_st;

参数：

Flag：1：开启 0：关闭

TimePeriod：布防时间段

\_\_pri\_TimePeriodCount：时间段个数

## IPCNetTimePeriod\_st

typedef struct //时间结构

{

[IPCNetTime\_st](#_IPCNetTime_st) Start;

[IPCNetTime\_st](#_IPCNetTime_st) End;

}IPCNetTimePeriod\_st;

参数：

Start：开始时间

End：结束时间

## IPCNetTime\_st

typedef struct //时间结构

{

int8\_t Hour;

int8\_t Min;

int8\_t Sec;

}IPCNetTime\_st;

参数：

Hour：时

Min：分

Sec：秒

## IPCNetVideoMoveInfo\_st

typedef struct //移动侦测区域设置

{

T\_S32 MdEnable;

T\_S32 Sensitive;

T\_S32 Columns;

T\_S32 Rows;

T\_S32 Area[IPCNET\_NET\_MAX\_AREA];

T\_S32 \_\_pri\_AreaCount;

}IPCNetVideoMoveInfo\_st;

参数：

MdEnable：1:开, 0:关

Sensitive：灵敏度1--9级，9级最敏感

Columns：行数

Rows：列数

Area[]：一般大小为Columns\*Rows

\_\_pri\_AreaCount：区域个数

## IPCNetMotionCellInfo\_st

typedef struct //移动侦测区域设置

{

T\_S32 MdEnable;

T\_S32 Sensitive;

T\_S32 Columns;

T\_S32 Rows;

char Cell[64];

}IPCNetMotionCellInfo\_st;

参数：

MdEnable：

Sensitive：0 - 10 灵敏度

Columns：矩阵行数

Rows：矩阵列数

Cell[]: 经过base64编码的Columns\*Rows的位图，给NULL默认全部置1

说明：

1. Columns\Rows: 是相对一个画面长宽的单位个数
2. Cell: 经过base64编码的Columns\*Rows的位图

## IPCNetAlarmLinkagePolicyInfo\_st

typedef struct

{

[IPCNetAlarmIoOutputInfo\_st](#_IPCNetAlarmIoOutputInfo_st) IoOutputInfo;

[IPCNetAlarmRecordInfo\_st](#_IPCNetAlarmRecordInfo_st) RecordInfo;

[IPCNetAlarmSnapInfo\_st](#_IPCNetAlarmSnapInfo_st) SnapInfo;

[IPCNetAlarmPresetInfo\_st](#_IPCNetAlarmPresetInfo_st) PresetInfo;

}IPCNetAlarmLinkagePolicyInfo\_st;

参数：

IoOutputInfo：告警联动IO信息

RecordInfo：报警录像信息

SnapInfo：报警抓拍信息

PresetInfo：报警后预置位动作

## IPCNetAlarmIoOutputInfo\_st

typedef struct //告警联动IO信息

{

int AlarmLevel;

int Delay;

int EmailEnable;

}IPCNetAlarmIoOutputInfo\_st;

参数：

AlarmLevel：0-检测到低电平报警，1-高电平报警

Delay：报警延时

EmailEnable：未实现

## IPCNetAlarmRecordInfo\_st

typedef struct //报警录像

{

T\_S32 RecEnable;

T\_S32 RecordTime;

T\_S32 FtpEnable;

T\_S32 EmailEnable;

T\_S32 CloudEnable;

}IPCNetAlarmRecordInfo\_st;

参数：

RecEnable：

RecordTime：

FtpEnable：Ftp显示 0-关闭 1-开启

EmailEnable：未实现

CloudEnable：云存储（未实现）

## IPCNetAlarmSnapInfo\_st

typedef struct //报警抓拍

{

T\_S32 PictureNum;

T\_S32 Sec;

T\_S32 Msec;

T\_S32 FtpEnable;

T\_S32 EmailEnable;

T\_S32 CloudEnable;

T\_S32 PushEnable;

}IPCNetAlarmSnapInfo\_st;

参数：

PictureNum：

Sec：

Msec：

FtpEnable：Ftp显示 0-关闭 1-开启

EmailEnable：未实现

CloudEnable：云存储（未实现）

PushEnable：报警上传 0-否 1-是

## IPCNetAlarmPresetInfo\_st

typedef struct //报警后预置位动作

{

T\_S32 PresetId;

char PresetName[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN   32)];

}IPCNetAlarmPresetInfo\_st;

参数：

PresetId：-1:表示没有设置关联预置点，>=0,设置了对应的预置点

PresetName[]：预置点名

## IpcnetNetworkWirelessSearch\_st

typedef struct

{

[IpcnetNetworkWirelessAp\_st](#_IpcnetNetworkWirelessAp_st) Aplist[30];//搜索WIFI信息

T\_S32 \_\_pri\_AplistCount;//ApCount

}IpcnetNetworkWirelessSearch\_st;

参数：

Aplist：热点信息

\_\_pri\_AplistCount：热点个数

## IPCNetEthConfig\_st

typedef struct

{

T\_S32 Valid;

T\_S32 DhcpEnble;

char Mac[20];

char IP[[IPCNET\_IP\_STRING\_LEN](#_IPCNET_IP_STRING_LEN     32)];

char Netmask[[IPCNET\_IP\_STRING\_LEN](#_IPCNET_IP_STRING_LEN    32)];

char Getway[[IPCNET\_IP\_STRING\_LEN](#_IPCNET_IP_STRING_LEN    32)];

char DNS1[[IPCNET\_IP\_STRING\_LEN](#_IPCNET_IP_STRING_LEN    32)];

char DNS2[[IPCNET\_IP\_STRING\_LEN](#_IPCNET_IP_STRING_LEN    32)];

}IPCNetEthConfig\_st;

参数：

Valid：

DhcpEnble：是否自动获取IP 0-否 1-是

Mac[]：not effective for setting

IP[]：IP

Netmask[]：子网掩码

Getway[]：网关

DNS1[]：DNS服务器1

DNS2[]：DNS服务器2

## Ipcnet3GInfo\_st

typedef struct

{

T\_S32 enable;

T\_S32 type;

char ip[32];

char vpn[32];

char name[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

}Ipcnet3GInfo\_st;

参数：

enable：使能3g模块，0-不生效，1-生效

type：3G模块类型，0-电信evdo,1-联通wcdma

ip：3G模块获取的IP地址

vpn：VPN

name：

## IPCNET\_WIRELESS\_STATUS\_e

typedef enum

{

//无线连接情况

IPCNET\_WIRELESS\_UNINITIALIZED,

IPCNET\_WIRELESS\_INITIALIZING,

IPCNET\_WIRELESS\_INTIALIZ\_FAIL, // 此状态出现在设备不存在或驱动有问题的情况，意味wireless设备不可用

IPCNET\_WIRELESS\_INTIALIZED,

IPCNET\_WIRELESS\_CONNECTING, // 4-进行热点连接中

IPCNET\_WIRELESS\_IP\_SETTING, // 5-热点连接成功，正在配制IP中，如果使用的是静态IP，则直接设置IP然后状态变为WIRELESS\_SETTING\_OK

IPCNET\_WIRELESS\_SETTING\_OK,

IPCNET\_WIRELESS\_SETTING\_FAIL, //设备连接或设置IP等操作失败

IPCNET\_WIRELESS\_PASSWD\_ERROR, //密码错误

}IPCNET\_WIRELESS\_STATUS\_e;

## IPCNetWirelessConfig\_st

typedef struct

{

T\_S32 ValidNetDev;

T\_S32 WirelessEnable;

T\_S32 WirelessStatus;

char SsidSetMode[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN       32)];

char EncType[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN       32)];

char SSID[[IPCNET\_WIFI\_AP\_MAX\_LEN](#_IPCNET_WIFI_AP_MAX_LEN  40)];

char Password[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

T\_S32 DhcpEnble;

char IP[[IPCNET\_IP\_STRING\_LEN](#_IPCNET_IP_STRING_LEN     32)];

char Netmask[[IPCNET\_IP\_STRING\_LEN](#_IPCNET_IP_STRING_LEN     32)];

char Getway[[IPCNET\_IP\_STRING\_LEN](#_IPCNET_IP_STRING_LEN     32)];

T\_S32 RSSI ;

}IPCNetWirelessConfig\_st;

参数：

ValidNetDev：当前使用网络设备：0-eth0,1-wireless,2-mobile //not effective for setting

WirelessEnable：

WirelessStatus：[IPCNET\_WIRELESS\_STATUS\_e](#_IPCNET_WIRELESS_STATUS_e) not effective for setting

SsidSetMode[]：not effective for setting

EncType[]：加密类型

SSID[]：WIFI名

Password[]：WIFI密码

DhcpEnble：自动获取IP

IP[]：IP

Netmask[]：子网掩码

Getway[]：网关

RSSI：信号强度假设60%这里就是60 not effective for setting

## IpcnetNetworkWirelessSearch\_st

typedef struct

{

[IpcnetNetworkWirelessAp\_st](#_IpcnetNetworkWirelessAp_st) Aplist[30];

T\_S32 \_\_pri\_AplistCount;

}IpcnetNetworkWirelessSearch\_st;

参数：

Aplist[]：搜索到的WIFI信息

\_\_pri\_AplistCount：WIFI个数

## IpcnetNetworkWirelessAp\_st

typedef struct

{

char SSID[[IPCNET\_WIFI\_AP\_MAX\_LEN](#_IPCNET_WIFI_AP_MAX_LEN  40)];

char EncType[[IPCNET\_STRING\_LEN](#_IPCNET_IP_STRING_LEN     32)];

T\_S32 RSSI;

}IpcnetNetworkWirelessAp\_st;

参数：

SSID[]：WIFI名称

EncType[]：加密方式

RSSI：信号强度假设60%这里就是60 not effective for setting

## IPCNetVideoEncodeCfg\_st

typedef struct

{

T\_S32 ViCh;

T\_S32 VideoEncodeNum;

[IPCNetVideoEncode\_st](#_IPCNetVideoEncode_st) VideoEncode[2];

int \_\_pri\_VideoEncodeCount;

}IPCNetVideoEncodeCfg\_st;

参数：

ViCh：镜头通道 ipc:0

VideoEncodeNum：

VideoEncode[]：视频编码参数

\_\_pri\_VideoEncodeCount：

## IPCNetVideoEncode\_st

typedef struct

{

T\_S32 EncCh;

char Encode[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

char OpionEncode[3][[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

T\_S32 \_\_pri\_OpionEncodeCount;

int Witdh;

int Height;

IPCNetVideoResolutionOpt\_st VideoResolutionOpt[4];

int \_\_pri\_VideoResolutionOptCount;

T\_S32 IFrame;

T\_S32 MaxIFrame;

T\_S32 MinIFrame;

T\_S32 FrameRate;

T\_S32 MaxFrameRate;

T\_S32 MinFrameRate;

T\_S32 RateCtrl;

T\_S32 Bitrate;

}IPCNetVideoEncode\_st;

参数：

EncCh：编码管道

Encode[]：

OpionEncode[][]：not effective for setting

\_\_pri\_OpionEncodeCount：not effective for setting

Witdh：

Height：

VideoResolutionOpt[]：not effective for setting

\_\_pri\_VideoResolutionOptCount：not effective for setting

IFrame：

MaxIFrame：

MinIFrame：

FrameRate：

MaxFrameRate：

MinFrameRate：

RateCtrl：CBR:1不可变码流,0:VBR可变码流

Bitrate：码率 16K-20M BIT/S为单位

## IpcnetAudioEncodeCfg\_st

typedef struct

{

T\_S32 AiCh;

T\_S32 AudioEncodeNum;

[IpcnetAudioEncode\_st](#_IpcnetAudioEncode_st) AudioEncde[1];//音频编码参数

T\_S32 \_\_pri\_AudioEncdeCount;//音频编码数

}IpcnetAudioEncodeCfg\_st;

参数：

AiCh：音频流通道

AudioEncodeNum：

AudioEncde[]：

\_\_pri\_AudioEncdeCount：

## IpcnetAudioEncode\_st

typedef struct

{

T\_S32 EncCh;

char Encode[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

char EncodeOption[4][[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

T\_S32 EncodeOptionCount;

char MicType[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

char MicTypeOption[4][[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

T\_S32 MicTypeOptionCount;

char SampleFreq[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

char SampleFreqOption[4][[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

T\_S32 SampleFreqOptionCount;

T\_S32 Vol;

T\_S32 MaxVol;

T\_S32 MinVol;

T\_S32 Gain;//sound

T\_S32 MaxGain;

T\_S32 MinGain;

T\_S32 DecVol;//speaker

T\_S32 DecMaxVol;

T\_S32 DecMinVol;

T\_S32 DecGain;

T\_S32 DecMaxGain;

T\_S32 DecMinGain;

}IpcnetAudioEncode\_st;

参数：

EncCh：编码通道

Encode[]：

EncodeOption[][]：

EncodeOptionCount：

MicType[]：

MicTypeOption[]：

MicTypeOptionCount：

SampleFreq[]：

SampleFreqOption[][]：

SampleFreqOptionCount：

//audio aenc vol and gain

Vol：

MaxVol：

MinVol：

Gain：

MaxGain：

MinGain：

//audio dec vol and gain

DecVol：

DecMaxVol：

DecMinVol：

DecGain：

DecMaxGain：

DecMinGain：

说明：

## IPCNetTimeCfg\_st

typedef struct

{

[IPCNetDate\_st](#_IPCNetDate_st) Date;

[IPCNetTime\_st](#_IPCNetTime_st_1) Time;

T\_S32 NtpEnable;

char NtpServ[128];

T\_S32 TimeZone; //IPCNET\_GMT\_e //旧协议使用该项，新协议使用DstDistId时，请将此项置-1

//new attr added

char DstDistId[64]; //夏令时地区ID，用字符串匹配

char ZoneDesc[32]; //新协议使用字符串来表示时区，作为获取时展示，设置时只使用DstDistId来确定

T\_S32 IsDst;

T\_S32 AdjustFlg;

T\_S32 DstEnable;

int TimeType;

}IPCNetTimeCfg\_st;

参数：

Date：日期

Time：时间

NtpEnable：

NtpServ[]：

TimeZone：

DstDistId[]：

ZoneDesc[]：

IsDst：当前该地区是否采用夏令时

AdjustFlg：是否已经校过时，用于测试两台设备时间相差的问题

DstEnable：自动调整夏令时开关

TimeType：0-格林威治时间，1-本地时间

## IPCNetAlarmMsgReport\_st

typedef struct

{

int AlarmType;

int Idx;

int Val;

int AlarmSta;

[IPCNetDate\_st](#_IPCNetDate_st) AlarmDate;

[IPCNetTime\_st](#_IPCNetTime_st_1) AlarmTime;

}IPCNetAlarmMsgReport\_st;

参数：

1. AlarmType：与 IPCNetAlarmTypeEnum\_st 对应
2. Idx：IPCNET\_ALARM\_IO\_INPUT报警时，idx为触发报警的序号
3. Val：IPCNET\_ALARM\_IO\_INPUT报警时，Val为触发报警的IO口的电平
4. AlarmSta：报警状态0-off,1-on
5. AlarmDate：报警日期
6. AlarmTime：报警时间

## IPCNetDate\_st

typedef struct //时间结构

{

int16\_t Year;

int8\_t Mon;

int8\_t Day;

int8\_t WDay;

}IPCNetDate\_st;

## IPCNetTime\_st

typedef struct //时间结构

{

int8\_t Hour;

int8\_t Min;

int8\_t Sec;

}IPCNetTime\_st;

## IPCNetRecordGetCfg\_st

typedef struct

{

T\_S32 ViCh;

char Path[[MAX\_PATH](#_MAX_PATH                   260)];

T\_S32 RecType;

}IPCNetRecordGetCfg\_st;

参数：

ViCh：sensor index

Path[]：录像路径

RecType：录像文件类型0-定时录像，1-报警录像

## IPCNetRecordCfg\_st

typedef struct

{

T\_S32 ViCh;

T\_S32 VeCh; //0-主码流,1-辅码流

BOOL AutoDel; //视频自动覆盖0-不覆盖，1-覆盖

T\_S32 RecMins;

T\_S32 RecMinsOption[8];

[IPCNetRecordTiming\_st](#_IPCNetRecordTiming_st) RecTime[8]; //录像时间

[IPCNetDiskInfo\_st](#_IPCNetDiskInfo_st) DiskInfo; //存储信息

T\_S32 PackageType;//0:avi,1:mjpg(vi.encode\_type == 1 is needed.),2:mkv

short Mode;//0:length means time, 1:length means file size.

//T\_S32 Duration;//if mode is 0,then 1000 means 00:10:00.

unsigned long long ReserveSize;

// #define RECYCLE\_MODE\_OVER\_WRITE 0

// #define RECYCLE\_MODE\_KEEP 1

//short RecycleMode;

}IPCNetRecordCfg\_st;

参数：

ViCh：

VeCh：

AutoDel：

RecMins：

RecMinsOption[]：

RecTime[]：

DiskInfo：

PackageType：

Mode：

ReserveSize：

## IPCNetRecordTiming\_st

typedef struct

{

T\_S32 En;

char St1[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

char Ed1[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

char St2[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

char Ed2[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

}IPCNetRecordTiming\_st;

参数：

St1[]：录像开始时间1

Ed1[]：录像结束时间1

St2[]：录像开始时间2

Ed2[]：录像结束时间2

## IPCNetDiskInfo\_st

typedef struct

{

char Path[[MAX\_PATH](#_MAX_PATH                   260)];//media file path the recorder to store.

BOOL isValid;

T\_S32 Type;//sdcard,hard disk,net disk or ?

unsigned Total;//mega bytes.

unsigned Free;//mega bytes.

}IPCNetDiskInfo\_st;

参数：

Path[]：

isValid：

Type：

Total：

Free：

## IPCPtzCtrlMsg\_st

typedef struct

{

[IPCNETPtzCmd\_e](#_IPCNETPtzCmd_e) CtrlCmd;

int Speed;

int RunTimes;

}IPCPtzCtrlMsg\_st;

参数：

CtrlCmd：控制命令

Speed：速度

RunTimes：巡航次数，巡航类的命令使用

## IPCNetCamInfo\_st

typedef struct

{

char DevName[64];

char Vendor[32];

char Model[32];

char Uuid[64];

char SysVer[32];

char FirmVer[32];

char SerialNo[64];

char IpcSdk[32];

char License[16];

T\_U8 alarm\_input\_num;

T\_U8 alarm\_output\_num;

T\_U8 tos;

T\_U8 hw\_num;

T\_U8 subdev\_num;

T\_U8 \_\_pri\_video\_inputCount;

[IpcnetAvInfo\_st](#_IpcnetAvInfo_st) video\_input[4];

}IPCNetCamInfo\_st;

参数：

DevName[]：设备名称

Vendor[]：厂商信息

Model[]：设备型号

Uuid[]：设备ID

SysVer[]：系统版本号

FirmVer[]：固件版本号custom.vendor.web.system.usr

SerialNo[]：设备序列号

IpcSdk[]：sdk版本号

License[]：校验码

alarm\_input\_num：

alarm\_output\_num：

tos：

hw\_num：

subdev\_num：

\_\_pri\_video\_inputCount：输入路数

video\_input[]：对应4个通道的内容，对于枪机只使用一个通道，回复的长度还是包括4个chann\_info，但只chann\_info[0]有效

## IpcnetAvInfo\_st

typedef struct IpcnetAvInfo

{

T\_U16 ch\_no;

char audio\_code\_type;

char audio\_sample\_freq;

T\_U16 width[2];

T\_U16 heigh[2];

char chann\_name[32];

}IpcnetAvInfo\_st;

参数：

ch\_no：通道号

audio\_code\_type：音频类型，枚举类型见下NPE\_AUDIO\_TYPE。 目前使用的音频类型有G726、G711A、G711U

audio\_sample\_freq：音频采样频率 ,枚举类型见下NPE\_AUDIO\_FREQ，目前固定使用8K

width[]：主辅两个通道的分辨率，width[0]、heigh[0]为主码流， width[1]、heigh[1]为辅码流

heigh[]：

chann\_name[]：通道名

## IPCNetDevName\_st

typedef struct

{

char DevName[128];

}IPCNetDevName\_st;

参数：

DevName[]：设备名

## IPCNetDiskInfo\_st

typedef struct

{

char Path[[MAX\_PATH](#_MAX_PATH                   260)];

BOOL isValid;

T\_S32 Type;//

unsigned Total;

unsigned Free;

}IPCNetDiskInfo\_st;

参数：

Path[]：media file path the recorder to store.

isValid：

Type：sdcard,hard disk,net disk or ?

Total：mega bytes.

Free：mega bytes

## IPCNetRet\_st

typedef struct

{

T\_S32 ret;

}IPCNetRet\_st;

参数：

ret：

## IPCNetCamColorCfg\_st

typedef struct

{

BOOL SetDefault; //是否设置默认值

T\_U8 ViCh;//max=3

T\_U8 Brightness;//[0-255] def:128

T\_U8 Chroma;//[0-255] def:128

T\_U8 Contrast;//[0-255] def:128

T\_U8 Saturtion;//[0-255] def:128

T\_U8 Acutance;//[0-255] def:128

}IPCNetCamColorCfg\_st;

参数：

SetDefault：

ViCh：

Brightness：

Chroma：

Contrast：

Saturtion：

Acutance：

## IPCNetVideoIFrame\_st

typedef struct

{

T\_S32 ViCh;

T\_S32 VenCh;

}IPCNetVideoIFrame\_st;

参数：

ViCh：视频流通道

VenCh：编码通道

## IPCNetGetOsdCfg\_st

typedef struct

{

int Vich;

}IPCNetGetOsdCfg\_st;

参数：

Vich：视频流通道

## IPCNetOsdCfg\_st

typedef struct

{

BOOL SetDefault;//

T\_U32 Vich;

[IPCNetOsdName\_st](#_IPCNetOsdName_st) OsdNameInfo; //

[IPCNetOsdConf\_st](#_IPCNetOsdConf_st) OsdDateInfo; //

[IPCNetOsdConf\_st](#_IPCNetOsdConf_st) OsdRateInfo; //

[IPCNetOsdConf\_st](#_IPCNetOsdConf_st) OsdSignalInfo;

}IPCNetOsdCfg\_st;

参数：

SetDefault：恢复默认值 true-是 false-否

Vich：视频流通道 ipc:0

OsdNameInfo：OSD设备名信息

OsdDateInfo：OSD日期信息

OsdRateInfo：OSD码率信息

OsdSignalInfo：

## IPCNetOsdName\_st

typedef struct

{

char NameText[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)];

[IPCNetOsdConf\_st](#_IPCNetOsdConf_st) NameConf;

}IPCNetOsdName\_st;

参数：

NameText[]：OSD名

NameConf：OSD配置信息

## IPCNetOsdConf\_st

typedef struct //通道时间显示基本信息

{

BOOL Enable; //是否启用

T\_U16 DisplayMode;

T\_U16 Xcord;

T\_U16 Ycord;

}IPCNetOsdConf\_st;

参数：

Enable：

DisplayMode：显示的格式，如2016/08/ 或者2016-08等

Xcord：x坐标0-704

Ycord：y坐标0-576

## IPCNetPicOverTurn\_st

typedef struct

{

T\_S32 ViCh;

BOOL Mirror;

BOOL Flip;

}IPCNetPicOverTurn\_st;

参数：

ViCh：视频流通道 ipc:0

Mirror：true-镜像，false-正常

Flip：true-翻转，false-正常

## IPCNETPrePointList\_st

typedef struct

{

unsigned int PointsCount;

IPCNETPointInfo\_st Points[32];

int \_\_pri\_PointsCount;

}IPCNETPrePointList\_st;

参数：

PointsCount：实际的预置点个数

Points[]：最多32个预置点

\_\_pri\_PointsCount：内部转json使用，不用管

## IPCNETPointInfo\_st

typedef struct

{

int BitID;

char Desc[32];

}IPCNETPointInfo\_st;

参数：

BitID：值为-1时，自动分配ID。>=0 指定ID

Desc[]：预置点名称

## IPCNETPointOpr\_st

typedef struct

{

unsigned int Type;

unsigned int BitID;

}IPCNETPointOpr\_st;

参数：

Type：0-调用预置点，1-清除单个预置点，2-清除所有预置点

BitID：预置点序号

## IPCNetFtpCfg\_st

typedef struct

{

char FtpAddr[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)]; //Ftp地址

T\_U32 FtpPort; //Ftp端口

char FtpUser[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)]; //Ftp用户

char FtpPasswd[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)]; //Ftp密码

char FtpPath[[IPCNET\_STRING\_LEN](#_IPCNET_STRING_LEN        32)]; //Ftp路径

}IPCNetFtpCfg\_st;

参数：

FtpAddr[]：

FtpPort：

FtpUser[]：

FtpPasswd[]：

FtpPath[]：

## IPCNetTalkReq\_t

typedef struct

{

int TalkEnable;

}IPCNetTalkReq\_t;

参数：

TalkEnable：是否开启speaker 0-关闭 1-开启

## IPCNetUpgradeInfo\_st

typedef struct

{

T\_S32 SerType;

[IpcnetUpgradeSystemInfo\_st](#_IpcnetUpgradeSystemInfo_st) UpgradeSystemInfo;

[IpcnetUpgradeCustomInfo\_st](#_IpcnetUpgradeCustomInfo_st) UpgradeCustomInfo;

[IpcnetUpgradeVendorInfo\_st](#_IpcnetUpgradeVendorInfo_st) UpgradeVendorInfo;

}IPCNetUpgradeInfo\_st;

参数：

SerType：0-远程服务器，1-本地服务器

UpgradeSystemInfo：

UpgradeCustomInfo：

UpgradeVendorInfo：厂商更新信息

## IpcnetUpgradeSystemInfo\_st

typedef struct

{

char version[32];

char usrcheck[32];

char systemcheck[32];

char webcheck[32];

}IpcnetUpgradeSystemInfo\_st;

参数：

version[]：

usrcheck[]：

systemcheck[]：

webcheck[]：

## IpcnetUpgradeCustomInfo\_st

typedef struct

{

char version[32];

char customcheck[32];

}IpcnetUpgradeCustomInfo\_st;

参数：

version[]：

customcheck[]：

## IpcnetUpgradeVendorInfo\_st

typedef struct

{

char version[32];

char vendorcheck[32];

}IpcnetUpgradeVendorInfo\_st;

参数：

version[]：

vendorcheck[]：

## IPCNetUpgradeCfg\_st

typedef struct

{

char LocalUrl[128];

char UpgradeUrl[128];

char SystemType[128];

char CustomType[128];

char VendorType[128];

[IPCNetAutoUpgrade\_st](#_IPCNetAutoUpgrade_st) AutoUpgrade;

}IPCNetUpgradeCfg\_st;

参数：

LocalUrl[]：局域网

UpgradeUrl[]：更新路径

SystemType[]：系统类型

CustomType[]：用户版本

VendorType[]：厂商版本

AutoUpgrade：自动更新设置

## IPCNetAutoUpgrade\_st

typedef struct

{

BOOL AutoUpgrade;

}IPCNetAutoUpgrade\_st;

参数：

AutoUpgrade：是否开启自动更新 true-是 false-否

## IPCNetAvRecordInfoReq\_st

typedef struct

{

int si; //

int m; //

char p[128]; //

int ds; //yyyymmdd 为方便录像是按时间查找而不是路径查找，当p为空时使用。属性代表开始时间:年月日

int de; //yyyymmdd 为方便录像是按时间查找而不是路径查找，当p为空时使用。属性代表结束时间:年月日

int et; //event type: 0-time; 1-alarm

int st; //start:hhmmss 开始时间:时分秒

int e; //end:hhmmss 结束时间:时分秒

}IPCNetAvRecordInfoReq\_st;

参数：

si：vich：视频流通道

m：mode 配合属性p使用，0:只查看文件，1:查看文件和文件夹

p[]：path

ds：

de：

et：

st：

e：

## IPCNetAvRecordInfoResp\_st

typedef struct

{

int n; //num

int t; //total space

int u; //free space

}IPCNetAvRecordInfoResp\_st;

参数：

n：

t：

u：

## IPCNetAvRecListPageReq\_t

typedef struct

{

char p[80];

int s; //start\_index

int c;

//新增 20170829

int si; //vich

int re; //recursive\_num

int m; //

int et;

}IPCNetAvRecListPageReq\_t;

参数：

p[]：

s：

c：count sdk限制了不能超过[IPCNET\_AV\_REC\_NUM](#_IPCNET_AV_REC_NUM        20)个

si：

re：

m：mode 0:不包括文件夹 1：包括文件夹

et：event type 0: 定时录像， 1：为报警录像

## IPCNetAvRecListPageResp\_t

typedef struct

{

int t;

char n[80];

[IPCNetAvRecFileInfo\_t](#_IPCNetAvRecFileInfo_t) l[[IPCNET\_AV\_REC\_NUM](#_IPCNET_AV_REC_NUM        20)];

int \_\_pri\_lCount;

}IPCNetAvRecListPageResp\_t;

参数：

t：DT\_DIR:文件夹，DT\_REG: 普通文件

n[]：dir name

l[]：文件信息

\_\_pri\_lCount：文件数目

## IPCNetAvRecFileInfo\_t

typedef struct

{

int t;

char n[24];

int sl;

int sh;

}IPCNetAvRecFileInfo\_t;

参数：

t：DT\_DIR:文件夹，DT\_REG: 普通文件

n[]：filename

sl：size 低8位

sh：size 高8位

## SNAP\_PAKT\_t

typedef struct

{

T\_U8 \*addr;

T\_U32 len;

}SNAP\_PAKT\_t;

参数：

addr：

len：

## SNAP\_FRAME\_t

typedef struct

{

int pkt\_num;

[SNAP\_PAKT\_t](#_SNAP_PAKT_t) \*pkt;

}SNAP\_FRAME\_t;

参数：

pkt\_num：

pkt：

# 回调

## WIFI热点搜索回调：SEARCH\_WIFI\_AP\_CB

typedef void (\*SEARCH\_WIFI\_AP\_CB)(

[IpcnetNetworkWirelessSearch\_st](#_IpcnetNetworkWirelessSearch_st) \*ap,

void \*user);

参数：

1. ap: 搜索到的热点
2. user: 用户自定义的用户数据

说明：

在此回调中处理所获取的WIFI信息

## SD卡格式化回调

typedef void (\*STORAGE\_FORMAT\_CB)(

[IPCNetRet\_st](#_IPCNetRet_st) \*pFomatResult,

void \*user);

参数：

1. pFomatResult：回调执行后从外部传进来的判断
2. user: 用户自定义的用户数据

说明：

在此函数中处理格式化信息

## 报警上报回调

## typedef void (\*IPCAM\_ALARM\_EVENT\_CB)(

[IPCNetAlarmMsgReport\_st](#_IPCNetAlarmMsgReport_st) \*alarm\_msg);

参数:

1. alarm\_msg报警上报后返回回来的信息

说明：

在此函数中处理报警信息

## 抓拍时间回调

typedef void (\*IPCAM\_SNAP\_PIC\_CB)(

[SNAP\_FRAME\_t](#_SNAP_FRAME_t) \*,

void \*);

参数：

说明：

在此函数中处理抓拍信息

人脸识别回调

typedef void (\*IPCAM\_FR\_EVENT\_CB)(IPCNetFrPicInfo\_st \*FrPicInfo,unsigned char \*pPicBuf);

参数：

typedef struct

{

IPCNetFrPicType\_e Type; //返回来的回调类型

int PicName[32]; //图片文字

int PicRemotePath[32]; //图片远程的路径

int PicLen; //图片长度byte

}IPCNetFrPicInfo\_st;

pPicBuf:图片数据

# 附加

### IPCNETPtzCmd\_e

typedef enum

{

IPCNET\_PTZ\_UP = 0,

IPCNET\_PTZ\_UP\_STOP,

IPCNET\_PTZ\_DOWN, // 2

IPCNET\_PTZ\_DOWN\_STOP,

IPCNET\_PTZ\_LEFT, // 4

IPCNET\_PTZ\_LEFT\_STOP,

IPCNET\_PTZ\_RIGHT, // 6

IPCNET\_PTZ\_RIGHT\_STOP,

IPCNET\_PTZ\_ZOOM\_IN, //8 放大

IPCNET\_PTZ\_ZOOM\_OUT, //9--缩小

IPCNET\_PTZ\_ZOOM\_STOP, // 10

IPCNET\_PTZ\_FOCUS\_FAR, // 11--远焦

IPCNET\_PTZ\_FOCUS\_NEAR, // 12 --近焦

IPCNET\_PTZ\_FOCUS\_STOP, //聚集停止

IPCNET\_PTZ\_SELF\_CHECK, //云台自检

IPCNET\_PTZ\_UP\_DOWN, //上下巡航

IPCNET\_PTZ\_LEFT\_RIGHT, //左右巡航

}IPCNETPtzCmd\_e;

# 宏定义

## IPCNET\_STRING\_LEN 32

## IPCNET\_IP\_STRING\_LEN 32

## IPCNET\_WIFI\_AP\_MAX\_LEN 40

## MAX\_PATH 260

## IPCNET\_AV\_REC\_NUM 20