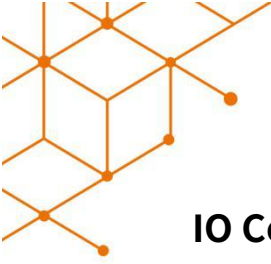


Kalay APP IO Command



IO Command 文档

简中版

版本号: V2.0.2

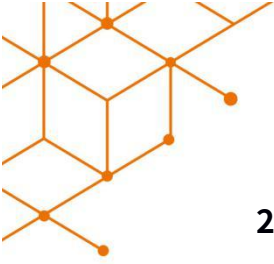
日期: 2021/6/29

作者: Lux

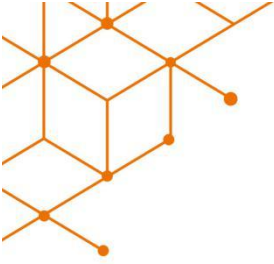
更新日期	版本号	更新内容	作者
2021/3/1	V2.0.0	初稿	Lux
2021/4/9	V2.0.1	修改 2.38 IOTYPE_USER_IPCAM_DEVICE_SUPPORT_CLOUD_RESP = 0x800D	Lux
2021/6/29	V2.0.2	增加 2.42 获取设备人形侦测开关 2.43 设置设备人形侦测开关 2.44 获取设备夜视开关 2.45 设置设备夜视开关 2.46 获取设备夏令时开关 2.47 设置设备夏令时开关 修改 2.7 获取设备事件列表	Lux

目录

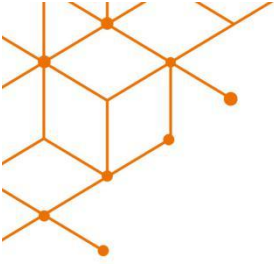
一、IO Command 定义.....	8
1.1 公版和客制化使用的参数区段.....	8
1.2 结构体定义规范及限制.....	8
二、结构体及说明.....	9
2.1 呼叫设备开始传送 Video Frame.....	9
IOTYPE_USER_IPCAM_START = 0x01FF;.....	9
2.2 呼叫设备停止传送 Video Frame.....	9
IOTYPE_USER_IPCAM_STOP = 0x02FF;.....	9



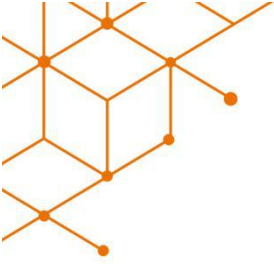
2.3 呼叫设备开始传送 Audio Frame.....	9
IOTYPE_USER_IPCAM_AUDIOSTART = 0x0300;.....	9
2.4 呼叫设备停止传送 Audio Frame.....	9
IOTYPE_USER_IPCAM_AUDIOSTOP = 0x0301;.....	9
2.5 设定设备 SD 卡录像模式.....	10
IOTYPE_USER_IPCAM_SETRECORD_REQ = 0x0310;.....	10
IOTYPE_USER_IPCAM_SETRECORD_RESP = 0x0311;.....	10
2.6 获取设备目前 SD 卡录像模式.....	10
IOTYPE_USER_IPCAM_GETRECORD_REQ = 0x0312;.....	10
IOTYPE_USER_IPCAM_GETRECORD_RESP = 0x0313;.....	11
2.7 获取设备事件列表.....	11
IOTYPE_USER_IPCAM_LISTEVENT_REQ = 0x0318;.....	11
IOTYPE_USER_IPCAM_LISTEVENT_RESP = 0x0319;.....	12
2.8 事件回播控制播放.....	13
IOTYPE_USER_IPCAM_RECORD_PLAYCONTROL_REQ = 0x031A;.....	13
IOTYPE_USER_IPCAM_RECORD_PLAYCONTROL_RESP = 0x031B;.....	14
2.9 设置设备解析度.....	14
IOTYPE_USER_IPCAM_SETSTREAMCTRL_REQ = 0x0320;.....	14
IOTYPE_USER_IPCAM_SETSTREAMCTRL_RESP = 0x0321;.....	14
2.10 获取设备目前解析度.....	15
IOTYPE_USER_IPCAM_GETSTREAMCTRL_REQ = 0x0322;.....	15
IOTYPE_USER_IPCAM_GETSTREAMCTRL_RESP = 0x0323;.....	15
2.11 设置设备位移侦测之灵敏度.....	15
IOTYPE_USER_IPCAM_SETMOTIONDETECT_REQ = 0x0324;.....	15
IOTYPE_USER_IPCAM_SETMOTIONDETECT_RESP = 0x0325;.....	16
2.12 获取设备目前位移侦测之灵敏度.....	16
IOTYPE_USER_IPCAM_GETMOTIONDETECT_REQ = 0x0326;.....	16
IOTYPE_USER_IPCAM_GETMOTIONDETECT_RESP = 0x0327;.....	16
2.13 获取目前设备通道数.....	17



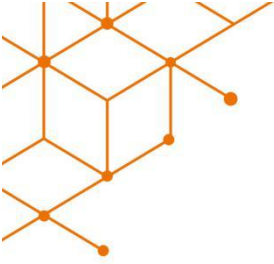
IOTYPE_USER_IPCAM_GETSUPPORTSTREAM_REQ = 0x0328;	17
IOTYPE_USER_IPCAM_GETSUPPORTSTREAM_RESP = 0x0329;	17
2.14 获取设备音讯格式（app 传送声音用）	17
IOTYPE_USER_IPCAM_GETAUDIOOUTFORMAT_REQ = 0x032A;	17
IOTYPE_USER_IPCAM_GETAUDIOOUTFORMAT_RESP = 0x032B;	18
2.15 获取设备信息（建议改为使用 0x8015/0x8016）	19
IOTYPE_USER_IPCAM_DEVINFO_REQ = 0x0330;	19
IOTYPE_USER_IPCAM_DEVINFO_RESP = 0x0331;	19
2.16 变更设备密码	19
IOTYPE_USER_IPCAM_SETPASSWORD_REQ = 0x0332;	19
IOTYPE_USER_IPCAM_SETPASSWORD_RESP = 0x0333;	19
2.17 事件回放进度控制	20
IOTYPE_USER_IPCAM_GET_PLAYBACK_REQ = 0x033A;	20
IOTYPE_USER_IPCAM_GET_PLAYBACK_RESP = 0x033B;	20
IOTYPE_USER_IPCAM_SET_RECORD_PROGRESS_REQ = 0x033C;	20
IOTYPE_USER_IPCAM_SET_RECORD_PROGRESS_RESP = 0x033D;	21
2.18 获取设备周围 Wifi 列表	21
IOTYPE_USER_IPCAM_LISTWIFIAP_REQ = 0x0340;	21
IOTYPE_USER_IPCAM_LISTWIFIAP_RESP = 0x0341;	21
2.19 设定设备的 Wifi 网络	22
IOTYPE_USER_IPCAM_SETWIFI_REQ = 0x0342;	22
IOTYPE_USER_IPCAM_SETWIFI_RESP = 0x0343;	22
2.20 获取设备目前所设置的 WiFi	23
IOTYPE_USER_IPCAM_GETWIFI_REQ = 0x0344;	23
IOTYPE_USER_IPCAM_GETWIFI_RESP = 0x0345;	23
2.21 设定设备目前所设置的 WiFi	23
IOTYPE_USER_IPCAM_SETWIFI_REQ2 = 0x0346;	23
IOTYPE_USER_IPCAM_GETWIFI_RESP2 = 0x0347;	24
2.22 呼叫设备开始接收 Audio Frame	24



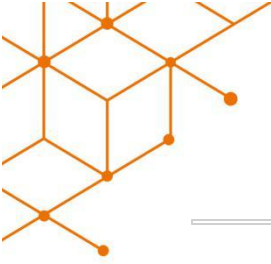
IOTYPE_USER_IPCAM_SPEAKERSTART = 0x0350;	24
2.23 呼叫设备停止接收 Audio Frame	24
IOTYPE_USER_IPCAM_SPEAKERSTOP = 0x0351;	24
2.24 设置画面镜像/翻转状态	25
IOTYPE_USER_IPCAM_SET_VIDEOMODE_REQ = 0x0370;	25
IOTYPE_USER_IPCAM_SET_VIDEOMODE_RESP = 0x0371;	25
2.25 获取画面镜像/翻转设置状态	25
IOTYPE_USER_IPCAM_GET_VIDEOMODE_REQ = 0x0372;	25
IOTYPE_USER_IPCAM_GET_VIDEOMODE_RESP = 0x0373;	26
2.26 格式化 SD 卡	26
IOTYPE_USER_IPCAM_FORMATTEXTSTORAGE_REQ = 0x0380;	26
IOTYPE_USER_IPCAM_FORMATTEXTSTORAGE_RESP = 0x0381;	26
2.27 获取 NVR 设备 Channel 接口数量	27
IOTYPE_USER_IPCAM_GET_NVR_CHANNEL_NUMBER_REQ = 0x5A4;	27
IOTYPE_USER_IPCAM_GET_NVR_CHANNEL_NUMBER_RESP = 0x5A5;	27
2.28 获取通道名称	27
IOTYPE_USER_IPCAM_GET_CHANNEL_NAME_REQ = 0x5B0;	27
IOTYPE_USER_IPCAM_GET_CHANNEL_NAME_RESP = 0x5B1;	28
2.29 设置通道名称	28
IOTYPE_USER_IPCAM_SET_CHANNEL_NAME_REQ = 0x5B2;	28
IOTYPE_USER_IPCAM_SET_CHANNEL_NAME_RESP = 0x5B3;	28
2.30 门铃呼叫 (Kalay2.0 不再使用)	28
IOTYPE_XM_CALL_REQ = 0x700;	28
IOTYPE_XM_CALL_RESP = 0x701;	29
IOTYPE_XM_CALL_IND = 0x702;	29
2.31 发送设备名称给设备	29
IOTYPE_USER_IPCAM_PUSH_DEVICENAME_REQ = 0x0736;	29
IOTYPE_USER_IPCAM_PUSH_DEVICENAME_RESP = 0x0737;	30
2.32 同步手机时间给设备	30



IOTYPE_USER_IPCAM_SET_TIME_SYNC_REQ = 0x0816;	30
IOTYPE_USER_IPCAM_SET_TIME_SYNC_RESP = 0x0817;	30
2.33 云台控制.....	31
IOTYPE_USER_IPCAM_PTZ_COMMAND = 0x1001;	31
2.34 APP 获取第一张 I 帧图片.....	32
IOTYPE_USER_IPCAM_RECEIVE_FIRST_IFRAME = 0x1002;	32
2.35 设备进行 OTA 升级.....	32
IOTYPE_USER_IPCAM_OTA_REQ = 0x8001;	32
IOTYPE_USER_IPCAM_OTA_RESP = 0x8002;	32
2.36 获取设备信息.....	33
IOTYPE_USER_IPCAM_DEVICE_INFO_REQ = 0x8015;	33
IOTYPE_USER_IPCAM_DEVICE_INFO_RESP = 0x8016;	33
2.37 获取设备是否支持 OTA 升级.....	33
IOTYPE_USER_IPCAM_DEVICE_SUPPORT_OTA_REQ = 0x800A;	34
IOTYPE_USER_IPCAM_DEVICE_SUPPORT_OTA_RESP = 0x800B;	34
2.38 获取设备是否支持云存储.....	34
IOTYPE_USER_IPCAM_DEVICE_SUPPORT_CLOUD_REQ = 0x800C;	34
IOTYPE_USER_IPCAM_DEVICE_SUPPORT_CLOUD_RESP = 0x800D;	34
2.39 设置设备云存储录像状态.....	35
IOTYPE_USER_IPCAM_DEVICE_SET_CLOUD_REQ = 0x8010;.....	35
IOTYPE_USER_IPCAM_DEVICE_SET_CLOUD_RESP = 0x8011;.....	35
2.40 获取设备云存储录像状态.....	35
IOTYPE_USER_IPCAM_DEVICE_GET_CLOUD_REQ = 0x8012;.....	35
IOTYPE_USER_IPCAM_DEVICE_GET_CLOUD_RESP = 0x8013;.....	35
2.41 获取有 SD 卡事件的日期.....	36
IOTYPE_USER_IPCAM_GET_EVENT_DATE_REQ = 0x9000;.....	36
IOTYPE_USER_IPCAM_GET_EVENT_DATE_RESP = 0x9001;.....	36
2.42 获取设备人形侦测开关.....	36
IOTYPE_USER_IPCAM_GET_HUMANDETECTION_REQ = 0x9002;.....	36



IOTYPE_USER_IPCAM_GET_HUMANDETECTION_RESP = 0x9003;.....	37
2.43 设置设备人形侦测开关.....	37
IOTYPE_USER_IPCAM_SET_HUMANDETECTION_REQ = 0x9004;.....	37
IOTYPE_USER_IPCAM_SET_HUMANDETECTION_RESP = 0x9005;.....	37
2.44 获取设备夜视开关.....	37
IOTYPE_USER_IPCAM_GET_NIGHTVISION_REQ = 0x9006;.....	37
IOTYPE_USER_IPCAM_GET_NIGHTVISION_RESP = 0x9007;.....	38
2.45 设置设备夜视开关.....	38
IOTYPE_USER_IPCAM_SET_NIGHTVISION_REQ = 0x9008;.....	38
IOTYPE_USER_IPCAM_SET_NIGHTVISION_RESP = 0x9009;.....	38
2.46 获取设备夏令时开关.....	38
IOTYPE_USER_IPCAM_GET_SUMMERTIME_REQ = 0x9010;.....	38
IOTYPE_USER_IPCAM_GET_SUMMERTIME_RESP = 0x9011;.....	39
2.47 设置设备夏令时开关.....	39
IOTYPE_USER_IPCAM_SET_SUMMERTIME_REQ = 0x9012;.....	39
IOTYPE_USER_IPCAM_SET_SUMMERTIME_RESP = 0x9013;.....	39
三、示例说明.....	40
3.1 APP 实作通过 Command 获取灯光状态实例.....	40
3.2 IOS 端发送及接收 Command 的方法.....	40
3.3 Android 如何将 bytes 转成一个 java 对象.....	40



一、IO Command 定义

注：设备需要依据 TUTK 提供的 IO Command 文档进行功能对接。为规范公版 Kalay APP IO Command 的设定和使用，禁止随意更改公版 IO Command 已制定的参数和结构体，如因随意更改造成设备无法成功对接公版 APP，TUTK 无需承担任何责任。同时，为满足客制化项目的需求，将预留部分参数区段以供客制化 Command 的设定。

1.1 公版和客制化使用的参数区段

公版使用参数区段	公版新增参数区段	客制化使用参数区段
0x1FF ~ 0x2FF; 0x300 ~ 0x3C3; 0x400 ~ 0x474; 0x500 ~ 0x5B9; 0x600 ~ 0x615; 0x700 ~ 0x747; 0x800 ~ 0x817; 0x1001 ~ 0x1002; 0x2001 ~ 0x224F; 0x4005 ~ 0x4015; 0x5001 ~ 0x5002; 0x6011 ~ 0x6018; 0x8000 ~ 0x8016; 0x9000 ~ 0x9013; 0x10004 ~ 0x1000B; 0x7F000000 ~ 0x7F000007; 0xFF001004 ~ 0xFF001008; 0x00000A03 ~ 0x00000A04; 0x00010000 ~ 0x00010003; 0x1FFF;	0x9000 ~ 0x9FFF; (之后新增请使用 0x9 开头表示 Kalay 公版新增的 command)	0x30000000 ~ 0x3000FFFF; (需以 0x3000 开头表示客制化新增的 command)

1.2 结构体定义规范及限制

- 结构体的数据所占字节数应为 4 的倍数
- 结构体的数据所占字节数应不超过 1024 字节



二、结构体及说明

2.1 呼叫设备开始传送 Video Frame

IOTYPE_USER_IPCAM_START = 0x01FF;

- 由 APP 发往 Device;
- APP 告知 Device 开始发送视频数据。
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;           // Camera Index
    unsigned char reserved[4];
} SMsgAVIoctlAVStream;
```

2.2 呼叫设备停止传送 Video Frame

IOTYPE_USER_IPCAM_STOP = 0x02FF;

- 由 APP 发往 Device;
- APP 告知 Device 停止发送视频数据。
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;           // Camera Index
    unsigned char reserved[4];
} SMsgAVIoctlAVStream;
```

2.3 呼叫设备开始传送 Audio Frame

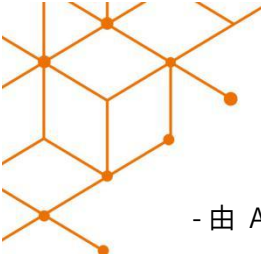
IOTYPE_USER_IPCAM_AUDIOSTART = 0x0300;

- 由 APP 发往 Device;
- APP 告知 Device 开始发送音频数据。
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;           // Camera Index
    unsigned char reserved[4];
} SMsgAVIoctlAVStream;
```

2.4 呼叫设备停止传送 Audio Frame

IOTYPE_USER_IPCAM_AUDIOSTOP = 0x0301;

- 
- 由 APP 发往 Device;
 - APP 告知 Device 停止发送音频数据。
 - IOCtrl 数据:

```
typedef struct
{
    unsigned int channel;    // Camera Index
    unsigned char reserved[4];
} SMsgAVIoctlAVStream;
```

2.5 设定设备 SD 卡录像模式

IOTYPE_USER_IPCAM_SETRECORD_REQ = 0x0310;

- 由 APP 发往 Device;
- APP 告知 Device 要设置设备 SD 卡录像模式。
- IOCtrl 数据:

```
typedef struct
{
    unsigned int channel;    // Camera Index
    unsigned int recordType; // 参考 ENUM_RECORD_TYPE
    unsigned char reserved[4];
} SMsgAVIoctlSetRecordReq, SMsgAVIoctlGetRecordResq;

typedef enum
{
    AVIOTC_RECORDTYPE_OFF = 0x00;
    AVIOTC_RECORDTYPE_FULLTIME = 0x01;
    AVIOTC_RECORDTYPE_ALARM = 0x02;
    AVIOTC_RECORDTYPE_MANUAL = 0x03;
} ENUM_RECORD_TYPE;
```

IOTYPE_USER_IPCAM_SETRECORD_RESP = 0x0311;

- 由 Device 发往 APP;
- Device 告知 App 设置结果。
- IOCtrl 数据:

```
typedef struct
{
    int result;    // 0 成功, otherwise: 失败。
    unsigned char reserved[4];
} SMsgAVIoctlSetRecordResp;
```

2.6 获取设备目前 SD 卡录像模式

IOTYPE_USER_IPCAM_GETRECORD_REQ = 0x0312;

- 由 APP 发往 Device;
- APP 告知 Device 要获取设备 SD 卡录像模式。
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;        // Camera Index
    unsigned char reserved[4];
}SMsgAVIoctlGetRecordReq;
```

IOTYPE_USER_IPCAM_GETRECORD_RESP = 0x0313;

- 由 Device 发往 APP;
- Device 将录像类型配置放到 IOCTL 资料并回传给 App。
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;        // Camera Index
    unsigned int recordType;     // 参考 ENUM_RECORD_TYPE
    unsigned char reserved[4];
}SMsgAVIoctlSetRecordReq, SMsgAVIoctlGetRecordResq;

typedef enum
{
    AVIOTC_RECORDTYPE_OFF = 0x00;
    AVIOTC_RECORDTYPE_FULLTIME = 0x01;
    AVIOTC_RECORDTYPE_ALARM = 0x02;
    AVIOTC_RECORDTYPE_MANUAL = 0x03;
}ENUM_RECORD_TYPE;
```

2.7 获取设备事件列表

IOTYPE_USER_IPCAM_LISTEVENT_REQ = 0x0318;

- 由 APP 发往 Device;
- APP 告知 Device 要获取设备事件列表。
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;        // Camera Index
    STimeDay stStartTime;       // 搜寻事件的起始时间 ...
    STimeDay stEndTime;        // ... 搜寻事件的结束时间
    unsigned char event;        // 事件类型, 参考 ENUM_EVENTTYPE
    unsigned char status;       // 0x00: 录像文件存在, 事件未读取
                                // 0x01: 录像文件存在, 事件已读取
                                // 0x02: 无录像文件记录
    unsigned char reserved[2];
```

```
}SMsgAVIoctlListEventReq;
```

```
typedef struct
```

```
{
```

```
    unsigned short year;    // 年份数
```

```
    unsigned char month;    // 自一月以来的月数，范围为 1 到 12。
```

```
    unsigned char day;      // 每月的日期，范围为 1 到 31。
```

```
    unsigned char wday;     // 自星期日以来的天数，范围为 0 到 6。（星期日=0，星期一=1，...）
```

```
    unsigned char hour;     // 午夜之后的小时数，范围为 0 到 23。
```

```
    unsigned char minute;   // 小时后的分钟数，范围为 0 到 59。
```

```
    unsigned char second;   // 分钟后的秒数，范围为 0 到 59。
```

```
}STimeDay;
```

IOTYPE_USER_IPCAM_LISTEVENT_RESP = 0x0319;

- 由 Device 发往 APP;
- Device 将录像列表放到 IOCTL 资料并回传给 App。
- IOCTL 数据:

```
typedef struct
```

```
{
```

```
    unsigned int channel;    // Camera Index
```

```
    unsigned int total;      // 搜索的总事件数
```

```
    unsigned char index;     // 数据包索引, 0,1,2...
```

```
                                // avSendIOCtrl 一次最多可发送 1024 个字节;
```

```
                                // 总事件可以分为 x 个包; x 是 0,1,2 ...
```

```
    unsigned char endflag;   // end flag = 1 表示此数据包是最后一个数据包，否则有更多的数据包要接收。
```

```
    unsigned char count;     // 此封包中有多少个事件。
```

```
    unsigned char reserved[1];
```

```
    SAvEvent stEvent[0];     // 此封包中所有事件的第一个地址
```

```
}SMsgAVIoctlListEventResp;
```

```
typedef struct
```

```
{
```

```
    STimeDay stTime;
```

```
    unsigned char event;     // 参考 ENUM_EVENTTYPE
```

```
    unsigned char status;    // 0x00: 录像文件存在，事件未读取
```

```
                                // 0x01: 录像文件存在，事件已读取
```

```
                                // 0x02: 无录像文件记录
```

```
    unsigned short duration; // 事件持续时间（以秒为单位）
```

```
}SAvEvent;
```

```
typedef enum
```

```
{
```

```
    AVIOCTRL_EVENT_ALL = 0x00;    // 所有事件类型
```

```
    AVIOCTRL_EVENT_MOTIONDECT = 0x01; // 移动侦测启动
```

```
    AVIOCTRL_EVENT_VIDEOLOST = 0x02; // 视频报警丢失
```

```
    AVIOCTRL_EVENT_IOALARM = 0x03; // IO 报警启动
```

```

AVIOCTRL_EVENT_MOTIONPASS = 0x04;    // 移动侦测结束
AVIOCTRL_EVENT_VIDEORESUME = 0x05;    // 视频检索
AVIOCTRL_EVENT_IOALARMPASS = 0x06;    // IO 报警结束
AVIOCTRL_EVENT_MOVIE = 0x07;
AVIOCTRL_EVENT_TIME_LAPSE = 0x08;
AVIOCTRL_EVENT_EMERGENCY = 0x09;
AVIOCTRL_EVENT_EXPT_REBOOT = 0x10;    // 系统异常重启
AVIOCTRL_EVENT_SDFAULT = 0x11;        // SD Card 记录异常
AVIOCTRL_EVENT_FULLTIME_RECORDING = 0x12; // 全时录像
AVIOCTRL_EVENT_PIR = 0x13;            // PIR 侦测
AVIOCTRL_EVENT_RINGBELL = 0x14;       // 门铃呼叫
AVIOCTRL_EVENT_SOUND = 0x15;
AVIOCTRL_EVENT_HUMANOID_DETECTION = 0x16; // 人形侦测
}ENUM_EVENTTYPE;

```

2.8 事件回播控制播放

IOTYPE_USER_IPCAM_RECORD_PLAYCONTROL_REQ = 0x031A;

-由 APP 发往 Device;

-APP 告知 Device 要进行录像事件回放。

-IOCtrl 数据:

```

typedef struct
{
    unsigned int channel;    // Camera Index
    unsigned int command;    // 播放记录指, 参考 ENUM_PLAYCONTROL.
    unsigned int Param;      // 指令参数, 由用户自定义
    STimeDay stTimeDay;      // 事件列表起始时间
    unsigned char endflag;    // 0 app, 事件播放完毕; 1 web, 当此事件播放完毕时继续播下一个事件
    unsigned char downloadFlag; // 0 回放模式, 1 下载模式
    unsigned char reserved[2];
} SMsgAVIoctrlPlayRecordReq;

typedef enum
{
    AVIOCTRL_RECORD_PLAY_PAUSE = 0x00;
    AVIOCTRL_RECORD_PLAY_STOP = 0x01;
    AVIOCTRL_RECORD_PLAY_STEPFORWARD = 0x02;
    AVIOCTRL_RECORD_PLAY_STEPBACKWARD = 0x03;
    AVIOCTRL_RECORD_PLAY_FORWARD = 0x04;
    AVIOCTRL_RECORD_PLAY_BACKWARD = 0x05;
    AVIOCTRL_RECORD_PLAY_SEEKTIME = 0x06;
    AVIOCTRL_RECORD_PLAY_END = 0x07;
    AVIOCTRL_RECORD_PLAY_START = 0x10;
    AVIOCTRL_RECORD_PLAY_NEXT = 0xf0;
    AVIOCTRL_RECORD_PLAY_IFRAME = 0xf1
}ENUM_PLAYCONTROL;

```



IOTYPE_USER_IPCAM_RECORD_PLAYCONTROL_RESP = 0x031B;

- 由 Device 发往 APP;
- Device 将录像回放结果放到 IOCtrl 资料并回传给 App。 (APP 收固定字节长度)
- IOCtrl 数据:

```
typedef struct
{
    unsigned int command;    // 播放记录指令，参考 ENUM_PLAYCONTROL
    unsigned int result;     // 取决于指令定义
                           // 当使用 AVIOCTRL_RECORD_PLAY_START:
                           // result >= 0 设备未使用实际通道进行播放
                           // result < 0 异常错误
                           // result = -1 回放异常
                           // result = -2 超出最大可连接的 Client 数

    unsigned int size;      // 事件回放文件大小

    unsigned char respond;
    unsigned char reserved[3];
} SMsgAVIOctrlPlayRecordResp;
```

2.9 设置设备解析度

IOTYPE_USER_IPCAM_SETSTREAMCTRL_REQ = 0x0320;

- 由 APP 发往 Device;
- APP 告知 Device 要设置设备的解析度。
- IOCtrl 数据:

```
typedef struct
{
    unsigned int channel;    // Camera Index
    unsigned char quality;   // 参考 ENUM_QUALITY_LEVEL
    unsigned char reserved[3];
} SMsgAVIOctrlSetStreamCtrlReq;

typedef enum
{
    AVIOCTRL_QUALITY_UNKNOWN = 0x00,
    AVIOCTRL_QUALITY_MAX = 0x01;
    AVIOCTRL_QUALITY_HIGH = 0x02;
    AVIOCTRL_QUALITY_MIDDLE = 0x03;
    AVIOCTRL_QUALITY_LOW = 0x04;
    AVIOCTRL_QUALITY_MIN = 0x05;
} ENUM_QUALITY_LEVEL;
```

IOTYPE_USER_IPCAM_SETSTREAMCTRL_RESP = 0x0321;

- 由 Device 发往 APP;

- 
- Device 告知 APP 解析度设置结果。
 - IOCTL 数据:

```
typedef struct
{
    unsigned int result;    // 0: 成功; otherwise: 失败
    unsigned char reserved[4];
} SMsgAVIoctlGetStreamCtrlResq;
```

2.10 获取设备目前解析度

IOTYPE_USER_IPCAM_GETSTREAMCTRL_REQ = 0x0322;

- 由 APP 发往 Device;
- APP 告知 Device 要获取设备目前的解析度。
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;    // Camera Index
    unsigned char reserved[4];
} SMsgAVIoctlGetStreamCtrlReq;
```

IOTYPE_USER_IPCAM_GETSTREAMCTRL_RESP = 0x0323;

- 由 Device 发往 APP;
- Device 将设备解析度配置放到 IOCTL 资料并回传给 App。
- IOCTL 数据:

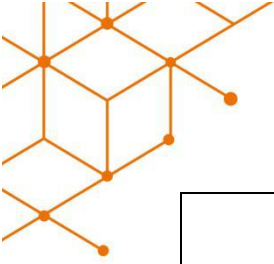
```
typedef struct
{
    unsigned int channel;    // Camera Index
    unsigned char quality;    //参考 ENUM_QUALITY_LEVEL
    unsigned char reserved[3];
} SMsgAVIoctlSetStreamCtrlReq, SMsgAVIoctlGetStreamCtrlResp;
```

2.11 设置设备位移侦测之灵敏度

IOTYPE_USER_IPCAM_SETMOTIONDETECT_REQ = 0x0324;

- 由 APP 发往 Device;
- APP 告知 Device 要设置设备位移侦测的灵敏度。
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;    // Camera Index
    unsigned int sensitivity;    // 0 (禁用) ~ 100(最大):
                                // Index in App. Sensitivity value
```



```
        // 0      0(关闭)
        // 1      25(低)
        // 2      50(中)
        // 3      75(高)
        // 4      100(最高)
    } SMsgAVIoctlSetMotionDetectReq, SMsgAVIoctlGetMotionDetectResp;
```

IOTYPE_USER_IPCAM_SETMOTIONDETECT_RESP = 0x0325;

- 由 Device 发往 APP;
- Device 告知 APP 位移侦测之灵敏度设置结果。
- IOCTL 数据:

```
typedef struct
{
    int result;           // 0: 成功; otherwise: 失败
    unsigned char reserved[4];
} SMsgAVIoctlSetMotionDetectResp;
```

2.12 获取设备目前位移侦测之灵敏度

IOTYPE_USER_IPCAM_GETMOTIONDETECT_REQ = 0x0326;

- 由 APP 发往 Device;
- APP 告知 Device 要获取设备位移侦测的灵敏度。
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel; // Camera Index
    unsigned char reserved[4];
} SMsgAVIoctlGetMotionDetectReq;
```

IOTYPE_USER_IPCAM_GETMOTIONDETECT_RESP = 0x0327;

- 由 Device 发往 APP;
- Device 将设备移动侦测灵敏度配置放到 IOCTL 资料并回传给 App。
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel; // Camera Index
    unsigned int sensitivity; // 0(禁用) ~ 100(最大):
                                // index sensitivity value
                                // 0      0(关闭)
                                // 1      25(低)
                                // 2      50(中)
                                // 3      75(高)
```




```
// 4          100(最高)
}SMsgAVIoctlSetMotionDetectReq, SMsgAVIoctlGetMotionDetectResp;
```

2.13 获取目前设备通道数

IOTYPE_USER_IPCAM_GETSUPPORTSTREAM_REQ = 0x0328;

- 由 APP 发往 Device;
- APP 告知 Device 要获取目前设备频道数。
- IOCTL 数据:

```
typedef struct
{
    unsigned char reserved[4];
}SMsgAVIoctlGetSupportStreamReq;
```

IOTYPE_USER_IPCAM_GETSUPPORTSTREAM_RESP = 0x0329;

- 由 Device 发往 APP;
- Device 将设备频道数配置放到 IOCTL 资料并回传给 App。
- IOCTL 数据:

```
typedef struct
{
    unsigned short index;      // 设备使用的 stream index
    unsigned short channel;    // AVAPIs 使用的 channel index
    char reserved[4];
}SStreamDef;

typedef struct
{
    unsigned int number;       // 可支持的 stream 数量
    SStreamDef streams[0];
}SMsgAVIoctlGetSupportStreamResp;
```

2.14 获取设备音讯格式 (app 传送声音用)

IOTYPE_USER_IPCAM_GETAUDIOOUTFORMAT_REQ = 0x032A;

- 由 APP 发往 Device;
- APP 告知 Device 要获取设备音频格式。
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;      // camera index
    char reserved[4];
}SMsgAVIoctlGetAudioOutFormatReq;
```

IOTYPE_USER_IPCAM_GETAUDIOOUTFORMAT_RESP = 0x032B;

- 由 Device 发往 APP;
- Device 将设备音频格式放到 IOCtrl 资料并回传给 App。
- IOCtrl 数据:

```
typedef struct
{
    unsigned int channel;    // camera index
    int format;              // 参考 ENUM_CODECID in AVFRAMEINFO.h
    char sample_rate;
    char bitdata;
    char channels;           // 可支持的通道数量
    char avservchannel;      // 0 子通道; 1 主通道; otherwise 子通道 (默认 0)
}SMMsgAVIOCtrlGetAudioOutFormatResp;
```

Sample_Rate:

```
AUDIO_SAMPLE_8K = 0;
AUDIO_SAMPLE_11K = 1;
AUDIO_SAMPLE_12K = 2;
AUDIO_SAMPLE_16K = 3;
AUDIO_SAMPLE_22K = 4;
AUDIO_SAMPLE_24K = 5;
AUDIO_SAMPLE_32K = 6;
AUDIO_SAMPLE_44K = 7;
AUDIO_SAMPLE_48K = 8;
```

CodecId:

```
MEDIA_CODEC_UNKNOWN = 0x00;
MEDIA_CODEC_VIDEO_MPEG4 = 0x4C;
MEDIA_CODEC_VIDEO_H263 = 0x4D;
MEDIA_CODEC_VIDEO_H264 = 0x4E;
MEDIA_CODEC_VIDEO_MJPEG = 0x4F;
MEDIA_CODEC_VIDEO_HEVC = 0x50;
MEDIA_CODEC_AUDIO_AAC_RAW = 0x86;
MEDIA_CODEC_AUDIO_AAC_ADTS = 0x87;
MEDIA_CODEC_AUDIO_AAC_LATM = 0x88;
MEDIA_CODEC_AUDIO_G711U = 0x89;    //g711 u-law
MEDIA_CODEC_AUDIO_G711A = 0x8A;    //g711 a-law
MEDIA_CODEC_AUDIO_ADPCM = 0x8B;
MEDIA_CODEC_AUDIO_PCM = 0x8C;
MEDIA_CODEC_AUDIO_SPEEX = 0x8D;
MEDIA_CODEC_AUDIO_MP3 = 0x8E;
MEDIA_CODEC_AUDIO_G726 = 0x8F;
```

Bitdata (0x0 ~ 0xF):

```
AUDIO_DATABITS_8 = 0;
AUDIO_DATABITS_16 = 1;
```



2.15 获取设备信息（建议改为使用 0x8015/0x8016）

IOTYPE_USER_IPCAM_DEVINFO_REQ = 0x0330;

- 由 APP 发往 Device;
- APP 告知 Device 要获取设备资讯。
- IOCTL 数据:

```
typedef struct
{
    unsigned char reserved[4];
}SMMsgAVIoctlDeviceInfoReq;
```

IOTYPE_USER_IPCAM_DEVINFO_RESP = 0x0331;

- 由 Device 发往 APP;
- Device 将设备资讯放到 IOCTL 资料并回传给 App。
- IOCTL 数据:

```
typedef struct
{
    unsigned char model[16];
    unsigned char vendor[16];
    unsigned int version;
    unsigned int channel;           //保留
    unsigned int total;            // MBytes, sdcard 的全部空间
    unsigned int free;            // MBytes , sdcard 的可用空间
    unsigned int company;         //参考 OTA 上的公司名
    unsigned char reserved[4];
}SMMsgAVIoctlDeviceInfoResp;
```

2.16 变更设备密码

IOTYPE_USER_IPCAM_SETPASSWORD_REQ = 0x0332;

- 由 APP 发往 Device;
- APP 告知 Device 要变更设备密码。
- IOCTL 数据:

```
typedef struct
{
    char oldpasswd[32];           // 旧密码
    char newpasswd[32];          // 新密码
}SMMsgAVIoctlSetPasswdReq;
```

IOTYPE_USER_IPCAM_SETPASSWORD_RESP = 0x0333;

- 由 Device 发往 APP;

- Device 告知 APP 变更设备密码的结果。
- IOCTL 数据:

```
typedef struct
{
    int result;           //0x00 密码设置成功, 否则失败
    unsigned char reserved[4];
}SMMsgAVIoctlSetPasswdResp;
```

2.17 事件回放进度控制

IOTYPE_USER_IPCAM_GET_PLAYBACK_REQ = 0x033A;

- 由 App 发往 Device。
- App 告知 Device 获取事件时长信息。
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;      // Camera Index
    STimeDay stTimeDay;       // 事件的时间长度
    unsigned char reserved[4];
}SMMsgAVIoctlGetPlaybackReq;
```

IOTYPE_USER_IPCAM_GET_PLAYBACK_RESP = 0x033B;

- 由 Device 发往 App。
- Device 时长信息回传给 App。
- Note:视频流的时间戳必须在回传的时间段内, 时间戳为 UTC 格式
- IOCTL 数据:

```
typedef struct
{
    unsigned int videoTime;    // 事件视频时间 (以秒为单位)
    unsigned long size;       // 事件大小 (以字节为单位)
    unsigned char reserved[4];
}SMMsgAVIoctlGetPlaybackResp;
```

IOTYPE_USER_IPCAM_SET_RECORD_PROGRESS_REQ = 0x033C;

- 由 App 发往 Device。
- App 告知 Device 从指定时间播放。
- Note: App 不会停掉视频流(不会发送 0x2ff)
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;      // Camera Index
```

```

STimeDay stTimeDay;          // 事件的时间长度
unsigned int progressTime;    // 事件进度时间（以秒为单位）
unsigned char reserved[4];
}SMMsgAVIoctlSetRecordProgressReq;

```

IOTYPE_USER_IPCAM_SET_RECORD_PROGRESS_RESP = 0x033D;

- 由 Device 发往 App。
- Device 将设备进度结果回传给 App。
- IOCTL 数据：

```

typedef struct
{
    unsigned char result;          //0 表示成功
    unsigned char reserved[3];
}SMMsgAVIoctlSeRecordProgressResp;

```

2.18 获取设备周围 Wifi 列表

IOTYPE_USER_IPCAM_LISTWIFIAP_REQ = 0x0340;

- 由 APP 发往 Device;
- APP 告知 Device 要获取设备周围的 wifi 列表。
- IOCTL 数据：

```

typedef struct
{
    unsigned char reserved[4];
}SMMsgAVIoctlListWifiApReq;

```

IOTYPE_USER_IPCAM_LISTWIFIAP_RESP = 0x0341;

- 由 Device 发往 APP;
- Device 将设备周围的 wifi 列表放到 IOCTL 资料并回传给 App。
- IOCTL 数据：

```

typedef struct
{
    unsigned int number;          // 最大: 1024/36=28
    SWifiAp stWifiAp[0];         // 存储所有 WiFi 信息的起始地
                                // 使用（SWifiAp）字节的大小来获取 WiFi 数据。
}SMMsgAVIoctlListWifiApResp;

typedef struct
{
    char ssid[32];               // WiFi SSID
    char mode;                   // 参考 ENUM_AP_MODE
    char enctype;                // 加密为 WiFi, 请参考 ENUM_AP_ENCTYPE。
    char signal;                 // 信号强度, 范围为 0 % 至 100 %。
}

```

```

char status;    // 0 : 无效的 ssid 或已断开连接
                // 1 : 连接默认网关
                // 2 : 密码不匹配
                // 3 : 弱信号且已连接
                // 4 : 所选密码匹配且已断开连接或已连接，但不是默认网关

}SWifiAp;

typedef enum
{
    AVIOTC_WIFIAPMODE_NULL = 0x00;
    AVIOTC_WIFIAPMODE_MANAGED = 0x01;
    AVIOTC_WIFIAPMODE_ADHOC = 0x02;
}ENUM_AP_MODE;

typedef enum
{
    AVIOTC_WIFIAPENC_INVALID = 0x00;
    AVIOTC_WIFIAPENC_NONE = 0x01;
    AVIOTC_WIFIAPENC_WEP = 0x02;           // WEP, 无密码
    AVIOTC_WIFIAPENC_WPA_TKIP = 0x03;
    AVIOTC_WIFIAPENC_WPA_AES = 0x04;
    AVIOTC_WIFIAPENC_WPA2_TKIP = 0x05;
    AVIOTC_WIFIAPENC_WPA2_AES = 0x06;
    AVIOTC_WIFIAPENC_WPA_PSK_TKIP = 0x07;
    AVIOTC_WIFIAPENC_WPA_PSK_AES = 0x08;
    AVIOTC_WIFIAPENC_WPA2_PSK_TKIP = 0x09;
    AVIOTC_WIFIAPENC_WPA2_PSK_AES = 0x0A;
}ENUM_AP_ENCTYPE;

```

2.19 设定设备的 Wifi 网络

IOTYPE_USER_IPCAM_SETWIFI_REQ = 0x0342;

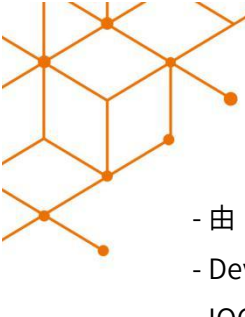
- 由 APP 发往 Device;
- APP 告知 Device 要设置设备的 wifi。
- IOCtrl 数据:

```

typedef struct
{
    unsigned char ssid[32];           // 连接的 WiFi SSID
    unsigned char password[32];       // WiFi SSID 密码
    unsigned char mode;               // 参考 ENUM_AP_MODE
    unsigned char enctype;            // 参考 ENUM_AP_ENCTYPE
    unsigned char reserved[10];
}SMMsgAVIOctrlSetWifiReq;

```

IOTYPE_USER_IPCAM_SETWIFI_RESP = 0x0343;

- 
- 由 Device 发往 APP;
 - Device 告知 APP 设置设备 wifi 的结果。
 - IOCtrl 数据:

```
typedef struct
{
    int result;           //如果已连接 WiFi，则返回 0，否则返回 1。
    unsigned char reserved[4];
}SMMsgAVIOctrlSetWifiResp;
```

2.20 获取设备目前所设置的 WiFi

IOTYPE_USER_IPCAM_GETWIFI_REQ = 0x0344;

- 由 APP 发往 Device;
- APP 告知 Device 要获取设备目前所设置的 wifi。
- IOCtrl 数据:

```
typedef struct
{
    unsigned char reserved[4];
}SMMsgAVIOctrlGetWifiReq;
```

IOTYPE_USER_IPCAM_GETWIFI_RESP = 0x0345;

- 由 Device 发往 APP;
- Device 将设备当前 wifi 配置放到 IOCtrl 资料并回传给 App。
- IOCtrl 数据:

```
typedef struct
{
    unsigned char ssid[32];           // WiFi ssid
    unsigned char password[32];       // WiFi 密码（如果不为空）
    unsigned char mode;               // 参考 ENUM_AP_MODE
    unsigned char enctype;            // 参考 ENUM_AP_ENCTYPE
    unsigned char signal;             // 信号强度 0--100%
    unsigned char status;             // 参考 SWifiAp 的 "status"
}SMMsgAVIOctrlGetWifiResp;
```

2.21 设定设备目前所设置的 WiFi

IOTYPE_USER_IPCAM_SETWIFI_REQ2 = 0x0346;

- 由 APP 发往 Device;
- APP 告知 Device 要设定设备目前所设置的 wifi。
- IOCtrl 数据:

```
typedef struct
```

```

{
    unsigned char ssid[32];           // 连接的 WiFi SSID
    unsigned char password[64];       // WiFi SSID 密码
    unsigned char mode;               // 参考 ENUM_AP_MODE
    unsigned char enctype;            // 参考 ENUM_AP_ENCTYPE
    unsigned char reserved[10];
}SMMsgAVIoctlSetWifiReq2;

```

注：该 command 支持 64bit 密码，同 IOTYPE_USER_IPCAM_SETWIFI_REQ = 0x0342 一起发送。

IOTYPE_USER_IPCAM_GETWIFI_RESP2 = 0x0347;

- 由 Device 发往 APP;
- Device 将设备当前 wifi 配置放到 IOCtrl 资料并回传给 App。
- IOCtrl 数据：

```

typedef struct
{
    unsigned char ssid[32];           // WiFi ssid
    unsigned char password[64];       // WiFi 密码（如果不为空）
    unsigned char mode;               // 参考 ENUM_AP_MODE
    unsigned char enctype;            // 参考 ENUM_AP_ENCTYPE
    unsigned char signal;             // 信号强度 0--100%
    unsigned char status;             // 参考 SWifiAp 的 "status"
}SMMsgAVIoctlGetWifiResp2;

```

2.22 呼叫设备开始接收 Audio Frame

IOTYPE_USER_IPCAM_SPEAKERSTART = 0x0350;

- 由 APP 发往 Device;
- APP 告知 Device 开始接收音频数据。
- IOCtrl 数据：

```

typedef struct
{
    unsigned int channel;              // Camera Index
    unsigned char reserved[4];
}SMMsgAVIoctlAVStream;

```

2.23 呼叫设备停止接收 Audio Frame

IOTYPE_USER_IPCAM_SPEAKERSTOP = 0x0351;

- 由 APP 发往 Device;
- APP 告知 Device 停止接收音频数据。

- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;    // Camera Index
    unsigned char reserved[4];
} SMsgAVIoctlAVStream;
```

2.24 设置画面镜像/翻转状态

IOTYPE_USER_IPCAM_SET_VIDEOMODE_REQ = 0x0370;

- 由 APP 发往 Device;

- APP 告知 Device 进行镜像/翻转设置。

- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;    // Camera Index
    unsigned char mode;      // 参考 ENUM_VIDEO_MODE
    unsigned char reserved[3];
} SMsgAVIoctlSetVideoModeReq;

typedef enum
{
    AVIOCTRL_VIDEOMODE_NORMAL = 0x00;
    AVIOCTRL_VIDEOMODE_FLIP = 0x01;        // 垂直翻转
    AVIOCTRL_VIDEOMODE_MIRROR = 0x02;      // 水平翻转
    AVIOCTRL_VIDEOMODE_FLIP_MIRROR = 0x03; // 垂直和水平翻转
} ENUM_VIDEO_MODE;
```

IOTYPE_USER_IPCAM_SET_VIDEOMODE_RESP = 0x0371;

- 由 Device 发往 APP;

- Device 告知 APP 设置结果。

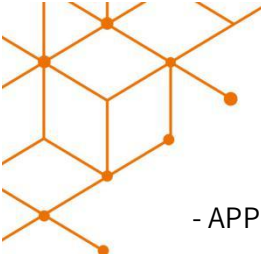
- IOCTL 数据:

```
typedef struct
{
    unsigned int channel;    // Camera Index
    unsigned char result;    // 0: 成功; otherwise:失败
    unsigned char reserved[3];
} SMsgAVIoctlSetVideoModeResp;
```

2.25 获取画面镜像/翻转设置状态

IOTYPE_USER_IPCAM_GET_VIDEOMODE_REQ = 0x0372;

- 由 APP 发往 Device;

- 
- APP 告知 Device 获取镜像/翻转设置状态。
 - IOCTL 数据：

```
typedef struct
{
    unsigned int channel;        // Camera Index
    unsigned char reserved[4];
}SMMsgAVIoctlGetVideoModeReq;
```

IOTYPE_USER_IPCAM_GET_VIDEOMODE_RESP = 0x0373;

- 由 Device 发往 APP;
- Device 告知 APP 目前画面镜像/翻转的设置状态。
- IOCTL 数据：

```
typedef struct
{
    unsigned int channel;        // Camera Index
    unsigned char mode;          // 参考 ENUM_VIDEO_MODE
    unsigned char reserved[3];
}SMMsgAVIoctlGetVideoModeResp;

typedef enum
{
    AVIOCTRL_VIDEOMODE_NORMAL = 0x00,
    AVIOCTRL_VIDEOMODE_FLIP = 0x01,          // 垂直翻转
    AVIOCTRL_VIDEOMODE_MIRROR = 0x02,        // 水平翻转
    AVIOCTRL_VIDEOMODE_FLIP_MIRROR = 0x03,   // 垂直和水平翻转
}ENUM_VIDEO_MODE;
```

2.26 格式化 SD 卡

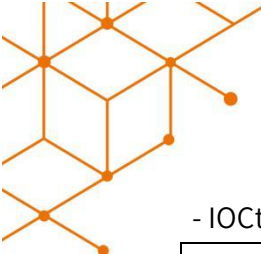
IOTYPE_USER_IPCAM_FORMATEXTSTORAGE_REQ = 0x0380;

- 由 APP 发往 Device;
- APP 告知 Device 执行格式化 SD 卡操作。
- IOCTL 数据：

```
typedef struct
{
    unsigned int storage;        // Storage index (例如 sdcard 插槽= 0, 内部闪存= 1, ...)
    unsigned char reserved[4];
}SMMsgAVIoctlFormatExtStorageReq;
```

IOTYPE_USER_IPCAM_FORMATEXTSTORAGE_RESP = 0x0381;

- 由 Device 发往 APP;
- Device 告知 APP 已格式化 SD 卡结果。



-IOCtrl 数据:

```
typedef struct
{
    unsigned int storage;    // Storage index
    char result;             // 0: 成功;
                           // -1:不支持格式化指令
                           // otherwise: 失败
    unsigned char reserved[3];
}SMMsgAVIoctlFormatExtStorageResp;
```

2.27 获取 NVR 设备 Channel 接口数量

IOTYPE_USER_IPCAM_GET_NVR_CHANNEL_NUMBER_REQ = 0x5A4;

- 由 APP 发往 Device;
- APP 告知 Device 获取设备 Channel 的接口数量。

-IOCtrl 数据:

```
typedef struct
{
    unsigned char reserved[4];
}SMMsgAVIoctlGetNVRChannelNumberReq;
```

IOTYPE_USER_IPCAM_GET_NVR_CHANNEL_NUMBER_RESP = 0x5A5;

- 由 Device 发往 APP;
- Device 告知 APP 目前设备 Channel 的接口数量。

-IOCtrl 数据:

```
typedef struct
{
    unsigned int number;    // 设备的通道数量
    unsigned char reserved[4];
}SMMsgAVIoctlGetNVRChannelNumberResp;
```

2.28 获取通道名称

IOTYPE_USER_IPCAM_GET_CHANNEL_NAME_REQ = 0x5B0;

- 由 APP 发往 Device;
- APP 告知 Device 获取设备的通道名称。

-IOCtrl 数据:

```
typedef struct
{
    unsigned char reserved[4];
}SMMsgAVIoctlGetChannelNameReq;
```



IOTYPE_USER_IPCAM_GET_CHANNEL_NAME_RESP = 0x5B1;

- 由 Device 发往 APP;
- Device 告知 APP 设备的通道名称。
- IOCTL 数据:

```
typedef struct
{
    unsigned char count;           // 通道数量
    unsigned char reserved[3];
    SChannelInfo sChannelInfo[0]; // 第一个 channelInfo 和 channelInfo 的总数
}SMsgAVIOctlGetChannelNameResp;

typedef struct
{
    unsigned int channel; //camera index
    unsigned char name[24]; //通道名称
} SChannelInfo;
```

2.29 设置通道名称

IOTYPE_USER_IPCAM_SET_CHANNEL_NAME_REQ = 0x5B2;

- 由 APP 发往 Device;
- APP 告知 Device 要设置设备的通道名称。
- IOCTL 数据:

```
typedef struct
{
    unsigned char count;           // SChannelInfos 的数量
    unsigned char reserved[3];
    SChannelInfo sChannelInfo[0]; // 第一个 channelInfo 和 channelInfo 的总数
}SMsgAVIOctlSetChannelNameReq;
```

IOTYPE_USER_IPCAM_SET_CHANNEL_NAME_RESP = 0x5B3;

- 由 Device 发往 APP;
- Device 告知 APP 设置设备通道名称的结果。
- IOCTL 数据:

```
typedef struct
{
    unsigned char result; // 0 成功, 其他失败
    unsigned char reserved[3];
}SMsgAVIOctlSetResetResp;
```

2.30 门铃呼叫 (Kalay2.0 不再使用)

IOTYPE_XM_CALL_REQ = 0x700;

- 由 Device 发往 APP;
- Device 告知 APP 有用户呼叫。
- IOCtrl 数据:-

```
typedef struct
{
    unsigned char index;        //门索引号, 0: 门 1; 1: 门 2
    STimeDay stTime;           //事件时间
    unsigned char reserved[3];
} SMsgAVIoctrlCallReq
```

~~IOTYPE_XM_CALL_RESP=0x701;~~

- 由 APP 发往 Device;
- APP 告知 Device 是否进行接听。
- IOCtrl 数据:-

```
typedef struct
{
    unsigned char index;        //门索引号, 0: 门 1; 1: 门 2
    int nAnswered;              //0: 挂断; 1: 接听
    unsigned char reserved[3];
} SMsgAVIoctrlCallResp;
```

~~IOTYPE_XM_CALL_IND=0x702;~~

- 由 Device 发往 APP;
- Device 告知 APP 是否有其他用户接听。
- IOCtrl 数据:-

```
typedef struct
{
    unsigned char index;        //门索引号, 0: 门 1; 1: 门 2
    unsigned char type;         //类型, 0: 有用户呼叫; 1: 有其他用户应答
    STimeDay stTime;           //事件时间
    unsigned char reserved[3];
} SMsgAVIoctrlCallInd;
```

2.31 发送设备名称给设备

IOTYPE_USER_IPCAM_PUSH_DEVICENAME_REQ=0x0736;

- 由 APP 发往 Device;
- APP 告知 Device 要当前设备名称。
- IOCtrl 数据:

```
typedef struct
{
```



```
unsigned int channel ;           //camera index
char devicename[150];           //当前设备名称
}SMsgAVIoctlPushDeviceNameReq;
```

IOTYPE_USER_IPCAM_PUSH_DEVICENAME_RESP= 0x0737;

- 由 Device 发往 APP;
- Device 告知 APP 要当前设备名称更新结果。
- IOCTL 数据:

```
typedef struct
{
    int res ;                //0:成功, 其他失败
    char reserved[4];
}SMsgAVIoctlPushDeviceNameResp;
```

2.32 同步手机时间给设备

IOTYPE_USER_IPCAM_SET_TIME_SYNC_REQ = 0x0816;

- 由 APP 发往 Device;
- APP 告知 Device 进行同步手机时间。
- IOCTL 数据:

```
typedef struct
{
    unsigned short year;
    unsigned char month;
    unsigned char day;
    unsigned char hour;
    unsigned char minute;
    unsigned char second;
    unsigned char nIsSupportSync;    // 1 支持; 0 不支持
    int nGMTOffset;                 // GMT 之间的偏移量 (以秒为单位)
} SMsgAVIoctlTimeSyncReq;
```

IOTYPE_USER_IPCAM_SET_TIME_SYNC_RESP = 0x0817;

- 由 Device 发往 APP;
- Device 告知 APP 同步时间的结果。
- IOCTL 数据:

```
typedef struct
{
    unsigned int result;           //如果成功则返回 0, 否则失败
    unsigned char reserved[4];
} SMsgAVIoctlTimeSyncResp;
```

2.33 云台控制

IOTYPE_USER_IPCAM_PTZ_COMMAND = 0x1001;

- 由 APP 发往 Device;
- APP 告知 Device 进行云台控制。
- IOCtrl 数据:

```
typedef struct
{
    unsigned char control;    // 云台控制指令，参考 ENUM_PTZCMD.
    unsigned char speed;     // 云台转动的速度
    unsigned char point;     // app 未使用
    unsigned char limit;     // app 未使用
    unsigned char aux;       // app 未使用
    unsigned char channel;   // camera index
    unsigned char reserve[2];
} SMsgAVIoctrlPtzCmd;
```

/* AVIOCTRL PTZ Command 值 */

```
typedef enum
{
    AVIOCTRL_PTZ_STOP = 0;
    AVIOCTRL_PTZ_UP = 1;
    AVIOCTRL_PTZ_DOWN = 2;
    AVIOCTRL_PTZ_LEFT = 3;
    AVIOCTRL_PTZ_LEFT_UP = 4;
    AVIOCTRL_PTZ_LEFT_DOWN = 5;
    AVIOCTRL_PTZ_RIGHT = 6;
    AVIOCTRL_PTZ_RIGHT_UP = 7;
    AVIOCTRL_PTZ_RIGHT_DOWN = 8;
    AVIOCTRL_PTZ_AUTO = 9;
    AVIOCTRL_PTZ_SET_POINT = 10;
    AVIOCTRL_PTZ_CLEAR_POINT = 11;
    AVIOCTRL_PTZ_GOTO_POINT = 12;
    AVIOCTRL_PTZ_SET_MODE_START = 13;
    AVIOCTRL_PTZ_SET_MODE_STOP = 14;
    AVIOCTRL_PTZ_MODE_RUN = 15;
    AVIOCTRL_PTZ_MENU_OPEN = 16;
    AVIOCTRL_PTZ_MENU_EXIT = 17;
    AVIOCTRL_PTZ_MENU_ENTER = 18;
    AVIOCTRL_PTZ_FLIP = 19;
    AVIOCTRL_PTZ_START = 20;
    AVIOCTRL_LENS_APERTURE_OPEN = 21;
    AVIOCTRL_LENS_APERTURE_CLOSE = 22;
    AVIOCTRL_LENS_ZOOM_IN = 23;
    AVIOCTRL_LENS_ZOOM_OUT = 24;
    AVIOCTRL_LENS_FOCAL_NEAR = 25;
```

```

AVIOCTRL_LENS_FOCAL_FAR = 26;
AVIOCTRL_AUTO_PAN_SPEED = 27;
AVIOCTRL_AUTO_PAN_LIMIT = 28;
AVIOCTRL_AUTO_PAN_START = 29;
AVIOCTRL_PATTERN_START = 30;
AVIOCTRL_PATTERN_STOP = 31;
AVIOCTRL_PATTERN_RUN = 32;
AVIOCTRL_SET_AUX = 33;
AVIOCTRL_CLEAR_AUX = 34;
AVIOCTRL_MOTOR_RESET_POSITION = 35;
}ENUM_PTZCMD;

```

2.34 APP 获取第一张 I 帧图片

IOTYPE_USER_IPCAM_RECEIVE_FIRST_IFRAME = 0x1002;

- 由 APP 发往 Device;
- APP 告知 Device 接收到第一张 I 帧图片。
- IOCtrl 数据:

```

typedef struct
{
    unsigned int channel;    // camera index
    char reserved[4];
}SMMsgAVIOctrlReceiveFirstIFrame;

```

2.35 设备进行 OTA 升级

IOTYPE_USER_IPCAM_OTA_REQ = 0x8001;

- 由 APP 发往 Device;
- APP 告知 Device 开始进行 OTA 升级。
- IOCtrl 数据:

```

typedef struct
{
    unsigned char file_checksum[32];    // MD5 校验码
    unsigned char url[256];            // 下载固件 url 链接
    unsigned int file_size;            // 固件包大小
    unsigned char reserved[4];
}SMMsgAVIOctrlOTAReq;

```

IOTYPE_USER_IPCAM_OTA_RESP = 0x8002;

- 由 Device 发往 App
- Device 告知 App 升级 OTA 结果。
- IOCtrl 数据:


```
typedef struct
{
    unsigned int progress;    // 下载固件进度
    unsigned int endflag;    //1: 下载完成; 0: 下载中
    unsigned char reserved[8];
}SMMsgAVIoctlOTAResp;
```

2.36 获取设备信息

IOTYPE_USER_IPCAM_DEVICE_INFO_REQ = 0x8015;

- 由 App 发往 Device
- App 告知 Device 要获取设备资讯。
- IOCTL 数据:

```
typedef struct
{
    unsigned char reserved[8];
} SMMsgAVIoctlDeviceInfoReq;
```

注：新设备需对接 OTA 功能时建议使用，与旧 cmd IOTYPE_USER_IPCAM_DEVINFO_REQ = 0x0330 同作用。

IOTYPE_USER_IPCAM_DEVICE_INFO_RESP = 0x8016;

- 由 Device 发往 App
- Device 告知 App 设备资讯。
- IOCTL 数据:

```
typedef struct
{
    unsigned char model[32];    // 产品型号
    unsigned char product[32]; // 产品名
    unsigned char vender[32];   // 制造商（需与 KOTA 服务器上的厂商一致）
    unsigned int version;       // 当前版本号
    unsigned int free;          // SD 卡剩余空间
    unsigned int total;         // SD 卡总空间
    unsigned char region;       // 获取设备所在位置，0：中国大陆区；1：非中国大陆区
    unsigned char reserved[3];
} SMMsgAVIoctlDeviceInfoResp;
```

注：以上参数用于拼接向 KOTA 服务器获取升级文件的下载地址请求，故需与在 KOTA 服务器上创建的 vender,product,model 等信息保持一致，且注意区分设备所在区域。

2.37 获取设备是否支持 OTA 升级



IOTYPE_USER_IPCAM_DEVICE_SUPPORT_OTA_REQ = 0x800A;

- 由 App 发往 Device
- App 告知 Device 要获取设备是否支持 OTA。
- IOCtrl 数据:

```
typedef struct
{
    unsigned char reserved[8];
} SMsgAVIoctrlDeviceSupportOTAReq;
```

IOTYPE_USER_IPCAM_DEVICE_SUPPORT_OTA_RESP = 0x800B;

- 由 Device 发往 App
- Device 告知 App 设备是否支持 OTA。
- IOCtrl 数据:

```
typedef struct
{
    unsigned char isSupport;    // 0 不支持; 1 支持
    unsigned char reserved[4];
} SMsgAVIoctrlDeviceSupportOTAResp;
```

2.38 获取设备是否支持云存储

IOTYPE_USER_IPCAM_DEVICE_SUPPORT_CLOUD_REQ = 0x800C;

- 由 App 发往 Device
- App 告知 Device 查询是否支持云存储。
- IOCtrl 数据:

```
typedef struct
{
    unsigned char reserved[8];
} SMsgAVIoctrlDeviceSupportCloudReq;
```

IOTYPE_USER_IPCAM_DEVICE_SUPPORT_CLOUD_RESP = 0x800D;

- 由 Device 发往 App
- Device 告知 App 设备是否支持云存储结果。
- IOCtrl 数据:

```
typedef struct
{
    unsigned int result;        // 0 不支持; 1 支持
    unsigned char reserved[4];
} SMsgAVIoctrlDeviceSupportCloudResp;
```

2.39 设置设备云存储录像状态

IOTYPE_USER_IPCAM_DEVICE_SET_CLOUD_REQ = 0x8010;

- 由 App 发往 Device
- App 告知 Device 开启或关闭云存储录像功能。
- IOCtrl 数据:

```
typedef struct
{
    unsigned int channel;    // camera index
    unsigned int isOn;      // 1 开启, 0 关闭
}SMMsgAVIoctlSetCloudReq;
```

IOTYPE_USER_IPCAM_DEVICE_SET_CLOUD_RESP = 0x8011;

- 由 Device 发往 App
- Device 告知 App 开启或关闭云存储录像结果。
- IOCtrl 数据:

```
typedef struct
{
    unsigned int channel;    // camera index
    unsigned int result;     // 0 成功, 1 失败
}SMMsgAVIoctlSetCloudResp;
```

2.40 获取设备云存储录像状态

IOTYPE_USER_IPCAM_DEVICE_GET_CLOUD_REQ = 0x8012;

- 由 App 发往 Device
- App 获取 Device 云存储录像状态。
- IOCtrl 数据:

```
typedef struct
{
    unsigned int channel;    // camera index
    unsigned char reserved[4];
}SMMsgAVIoctlGetCloudReq;
```

IOTYPE_USER_IPCAM_DEVICE_GET_CLOUD_RESP = 0x8013;

- 由 Device 发往 App
- Device 告知 App 云存储录像状态。
- IOCtrl 数据:

```
typedef struct
{
    unsigned int channel;    // camera index
```



```
unsigned int isOn;      // 1 开启, 0 关闭
} SMsgAVIoctlGetCloudResp;
```

2.41 获取有 SD 卡事件的日期

IOTYPE_USER_IPCAM_GET_EVENT_DATE_REQ = 0x9000;

- 由 App 发往 Device
- App 告知 Device 获取有 SD 卡事件的日期。
- IOCtrl 数据:

```
typedef struct
{
    unsigned int channel;      //Camera Index
    unsigned int eventType;    //查询的事件类型, 0: 全部, 1: 移动侦测; 2: 全时录像, 默认是 0
    STimeDay stStartTime;     //开始时间
    STimeDay stEndTime;       //结束时间
    unsigned char reserved[8];
} SMsgAVIoctlGetEventDateReq;
```

IOTYPE_USER_IPCAM_GET_EVENT_DATE_RESP = 0x9001;

- 由 Device 发往 App
- Device 告知 App 有 SD 卡事件的日期结果。
- IOCtrl 数据:

```
typedef struct
{
    unsigned int count;        //事件的日期数量
    unsigned char reserved[4];
    EventDate eventDate[1];    //事件的日期
} SMsgAVIoctlGetEventDateResp;

typedef struct
{
    char date[8];              //事件的日期, 例如: 20200210, 后面依次还有 count-1 个 date[8]
} EventDate;
```

2.42 获取设备人形侦测开关

IOTYPE_USER_IPCAM_GET_HUMANDETECTION_REQ = 0x9002;

- 由 App 发往 Device
- App 告知 Device 查询人形侦测开关
- IOCtrl 数据:

```
typedef struct
{
    unsigned char reserved[4];
}
```



```
} SMsgAVIoctrlGetHumanDetectionReq;
```

IOTYPE_USER_IPCAM_GET_HUMANDETECTION_RESP = 0x9003;

- 由 Device 发往 App
- Device 告知 App 人形侦测开关查询结果
- IOCtrl 数据:

```
typedef struct
{
    unsigned int result;        // 1:开启; 其他:关闭
    unsigned char reserved[4];
} SMsgAVIoctrlGetHumanDetectionResp;
```

2.43 设置设备人形侦测开关

IOTYPE_USER_IPCAM_SET_HUMANDETECTION_REQ = 0x9004;

- 由 App 发往 Device
- App 告知 Device 设置人形侦测开关
- IOCtrl 数据:

```
typedef struct
{
    unsigned int isOn;         // 1:开启; 0:关闭
    unsigned char reserved[4];
} SMsgAVIoctrlSetHumanDetectionReq;
```

IOTYPE_USER_IPCAM_SET_HUMANDETECTION_RESP = 0x9005;

- 由 Device 发往 App
- Device 告知 App 人形侦测开关设置结果
- IOCtrl 数据:

```
typedef struct
{
    unsigned int result;        // 0:成功; 其他:失败
    unsigned char reserved[4];
} SMsgAVIoctrlSetHumanDetectionResp;
```

2.44 获取设备夜视开关

IOTYPE_USER_IPCAM_GET_NIGHTVISION_REQ = 0x9006;

- 由 App 发往 Device
- App 告知 Device 查询夜视开关
- IOCtrl 数据:



```
typedef struct
{
    unsigned char reserved[4];
} SMsgAVIoctrlGetNightVisionReq;
```

IOTYPE_USER_IPCAM_GET_NIGHTVISION_RESP = 0x9007;

- 由 Device 发往 App
- Device 告知 App 夜视开关查询结果
- IOCtrl 数据:

```
typedef struct
{
    unsigned int result;        // 1:开启; 其他:关闭
    unsigned char reserved[4];
} SMsgAVIoctrlGetNightVisionResp;
```

2.45 设置设备夜视开关

IOTYPE_USER_IPCAM_SET_NIGHTVISION_REQ = 0x9008;

- 由 App 发往 Device
- App 告知 Device 设置夜视开关
- IOCtrl 数据:

```
typedef struct
{
    unsigned int isOn;        // 1:开启; 0:关闭
    unsigned char reserved[4];
} SMsgAVIoctrlSetNightVisionReq;
```

IOTYPE_USER_IPCAM_SET_NIGHTVISION_RESP = 0x9009;

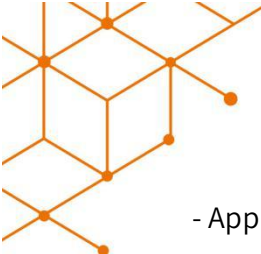
- 由 Device 发往 App
- Device 告知 App 夜视开关设置结果
- IOCtrl 数据:

```
typedef struct
{
    unsigned int result;        // 0:成功; 其他:失败
    unsigned char reserved[4];
} SMsgAVIoctrlSetNightVisionResp;
```

2.46 获取设备夏令时开关

IOTYPE_USER_IPCAM_GET_SUMMERTIME_REQ = 0x9010;

- 由 App 发往 Device

- 
- App 告知 Device 查询夏令时开关
 - IOCtrl 数据:

```
typedef struct
{
    unsigned char reserved[4];
} SMsgAVIoctrlGetSummerTimeReq;
```

IOTYPE_USER_IPCAM_GET_SUMMERTIME_RESP = 0x9011;

- 由 Device 发往 App
- Device 告知 App 夏令时开关查询结果
- IOCtrl 数据:

```
typedef struct
{
    unsigned int result;          // 1:开启; 其他:关闭
    unsigned char reserved[4];
} SMsgAVIoctrlGetSummerTimeResp;
```

2.47 设置设备夏令时开关

IOTYPE_USER_IPCAM_SET_SUMMERTIME_REQ = 0x9012;

- 由 App 发往 Device
- App 告知 Device 设置夏令时开关
- IOCtrl 数据:

```
typedef struct
{
    unsigned int isOn;           // 1:开启; 0:关闭
    unsigned char reserved[4];
} SMsgAVIoctrlSetSummerTimeReq;
```

IOTYPE_USER_IPCAM_SET_SUMMERTIME_RESP = 0x9013;

- 由 Device 发往 App
- Device 告知 App 夏令时开关设置结果
- IOCtrl 数据:

```
typedef struct
{
    unsigned int result;          // 0:成功; 其他:失败
    unsigned char reserved[4];
} SMsgAVIoctrlSetSummerTimeResp;
```

三、示例说明

3.1 APP 实作通过 Command 获取灯光状态实例

- 定义 Command 及结构体
- 获取开关状态 Command 定义：IOTYPE_GET_LED_REQ = 0x30000001
- 获取开关状态结构体定义：@struct SMsgAVIoctlGetLedReq;

```
typedef struct
{
    int channel;    //当前通道号
    unsigned char reserved[4];
}SMsgAVIoctlGetLedReq;
```

- 设备端回复 APP 开关状态 Command 定义：IOTYPE_GET_LED_RESP = 0x30000002
- 设备端回复 APP 开关状态结构体定义：@struct SMsgAVIoctlGetLedResp;

```
typedef struct
{
    int res;        //0: 成功, 其他失败
    unsigned char isOnOff;    // 0:开启; 1:关闭
    unsigned char reserved[3];
}SMsgAVIoctlGetLedResp;
```

3.2 IOS 端发送及接收 Command 的方法

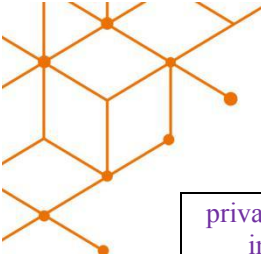
- 发送 Command 的方法

```
SMsgAVIoctlGetLedReq *s = malloc(sizeof(SMsgAVIoctlGetLedReq));
memset(s, 0, sizeof(SMsgAVIoctlGetLedReq));
[self.camera KY_SendIOCtrlToChannel:channel
    Type:IOTYPE_GET_LED_REQ
    Data:(char *)s
    DataSize:sizeof(SMsgAVIoctlGetLedReq)];
free(s);
```

- 接收 Command 的回调方法

```
-(void)KY_DidReceiveIOCtrlWithUid:(NSString *)uid Type:(NSInteger)type Data:(const char*)data
DataSize:(NSInteger)size
{
    if (type == IOTYPE_GET_LED_RESP)
    {
        SMsgAVIoctlGetLedResp *s = (SMsgAVIoctlGetLedResp *)data;
        isLightOn = s->isOnOff;
    }
}
```

3.3 Android 如何将 bytes 转成一个 java 对象



```

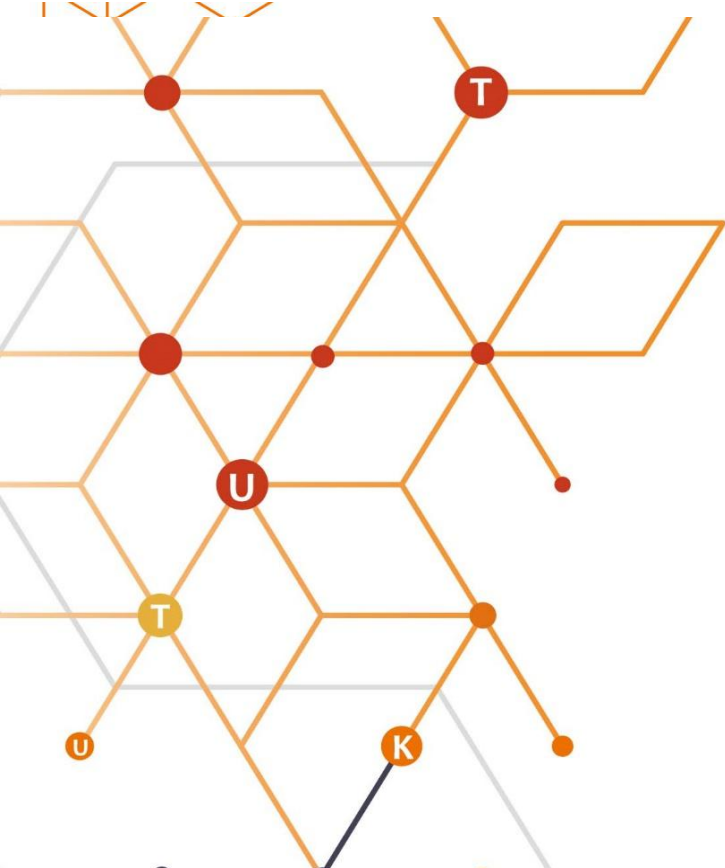
private class People {
    int age = 0;//4 bytes means data[0] ~ data[3]
    long birthday = 1612330883993L;//8 bytes means data[4] ~ data[11]
    String name = "";//custom bytes ex: 32 bytes
}
/**
 * byte to people
 *
 * @param data src bytes length > 44
 * @return people
 */
private People byteToPeople(byte[] data) {
    if (data.length < 44) {
        return null;
    }
    People p = new People();
    p.age = byteArrayToInt(data, 0);
    p.birthday = byteArrayToLong(data, 4);
    byte[] name = new byte[32];
    System.arraycopy(data, 12, name, 0, 32);
    p.name = new String(name);
    return p;
}

private short byteArrayToShort(byte[] bytes, int beginPos) {
    return (short) (((0xff & bytes[beginPos]) | ((0xff & bytes[beginPos + 1]) << 8));
}

private int byteArrayToInt(byte[] bytes, int beginPos) {
    return (0xff & bytes[beginPos]) | (0xff & bytes[beginPos + 1]) << 8 | (0xff & bytes[beginPos + 2]) << 16 |
    (0xff & bytes[beginPos + 3]) << 24;
}

private long byteArrayToLong(byte[] bytes, int beginPos) {
    long l = 0;
    for (int i = 0; i < 4; i++) {
        l = l | ((0xffL & bytes[beginPos + i]) << (8 * i));
    }
    return l;
}

```



Pioneering M2M Solutions

www.thoughtek.com

9F, No. 364, Sec. 1, Nangang Rd.,
Nangang Dist., Taipei City 11579,
Taiwan

+886-2-2653-5111