# 协议包结构

请求：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段 | 起始符 | 校验 | 长度 | 包类型 | 版本号 | 序列号SN | 时间 | 内容 | 结束符 |
| 长度  (字节) | 2 | 2 | 2 | 1 | 1 | 4 | 6 | N | 2 |

响应：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段 | 起始符 | 校验 | 长度 | 包类型 | 版本号 | 序列号SN | 时间 | 内容 | 结束符 |
| 长度  (字节) | 2 | 2 | 2 | 1 | 1 | 4 | 6 | N | 2 |

1. 起始：0xFFFF
2. 校验：从长度到内容结束，不包含结束符。采用crc16校验算法
3. 长度：从起始到结束符的所有长度
4. 包类型：消息类型
5. 版本号：每个消息的版本，根据这个后期可以支持消息类型的结构变更和调整
6. 序列号：开机从0开始，每一个数据包递增
7. 时间：年月日时分秒 ，6个字节
8. 内容：每个消息类型有自己独立的数据内容
9. 结束符：0xEEEE

# 规则约定

1. 请求是奇数，回复加1是偶数
2. 每个数据包需要携带发送方的UID（unique ID）
3. 响应中的sn是将请求中的sn原样返回。
4. 时间为发送方本地RTC时钟，如果本地没有RTC，可以留空

# ACK 包

1. 收到任何一个包都先回ACK包，再做约定的业务逻辑。
2. ACK的消息类型为固定值（0XFF），其他遵循上面的约定。设备端看到这个特定的类型就知道是ACK包，不会产生错误的嵌套处理。

# 协议内容

## Beacon控制包

### 状态beacon包 0x01

请求：

|  |  |
| --- | --- |
| 字段 | 状态 |
| 长度  (字节) | 1 |

响应：无

1. 状态：00：正常，01：未定义, 02：报警,04：高空坠落 08：正在取消

目前WIfi Beacon包中Payload部分数据格式：

C语言结构体：

typedef struct

{

uint8\_t TimeStamper[8];

uint8\_t BeaconInter[2];

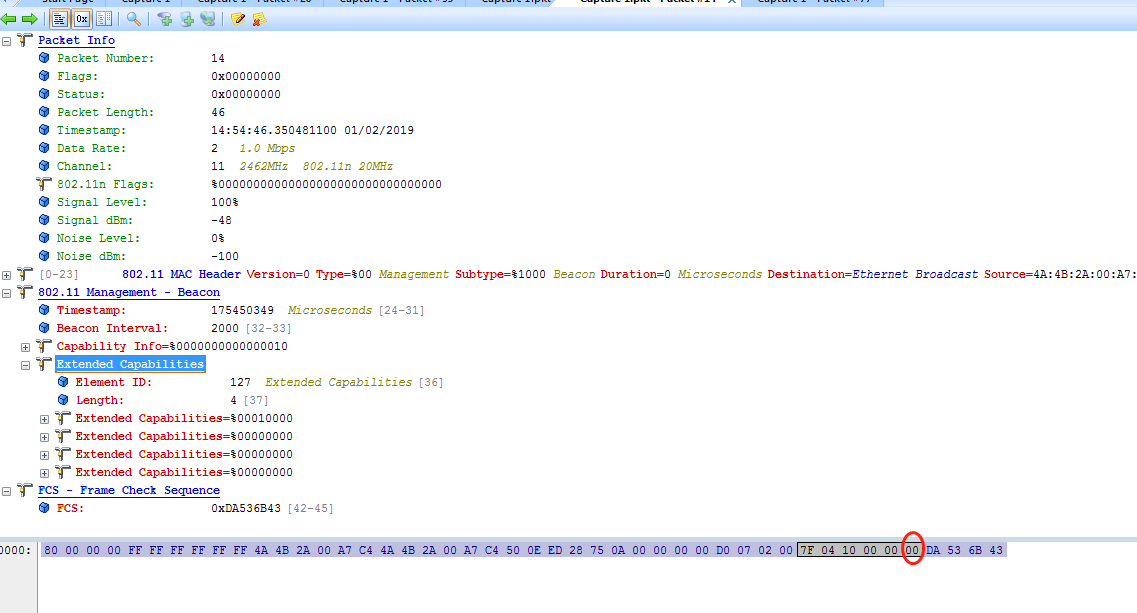
uint8\_t CapabilityInf[2];

uint8\_t ExtendCap[6];

} payload\_struct;

目前我们用到的是ExtendCap[5]，00：正常，01,02：报警,04：高空坠落分别代表不同含义。

对应的抓包工具抓出来的格式如下：



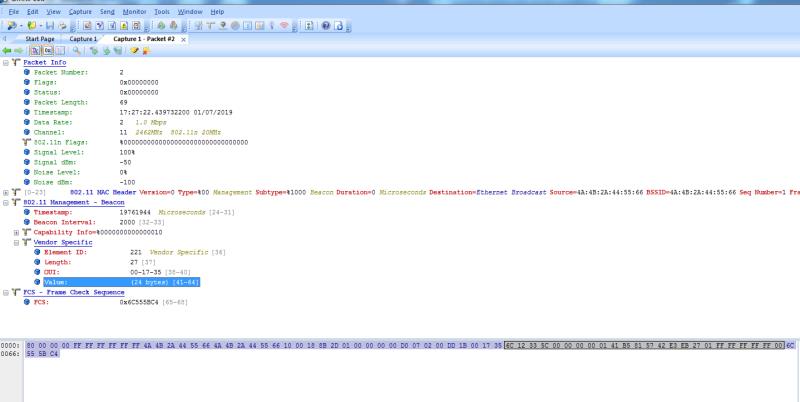
### GPS beacon包 0x03

请求：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段 | 时间戳 | 时间属性 | 纬度 | 经度 | 状态 | 保留 |
| 长度  (字节) | 8 | 1 | 4 | 4 | 1 | 6 |

响应：无

1. 时间戳：000000005C33126C,为时间戳，换算为RTC时间为2019-01-07 16:48:44
2. 时间属性：01：历史帧，（00代表即时帧）
3. 纬度：5781B541，为纬度，换算为浮点数为22.688155
4. 经度：27EBE342，为经度，换算为浮点数为113.959283
5. 状态：02为报警状态（还有04-高空坠落，00-正常定位帧）
6. 保留：后面6个字节为保留字段



顺序是时间（8字节）、历史帧即时帧标识（1字节）、纬度（4字节）、经度（4字节）、状态标识（1字节）

阴影选中部分：

000000005C33126C,为时间戳，换算为RTC时间为2019-01-07 16:48:44

01：历史帧，（00代表即时帧）

5781B541，为纬度，换算为浮点数为22.688155

27EBE342，为经度，换算为浮点数为113.959283

01为报警状态（还有04-高空坠落，00-正常定位帧）

后面6个字节为保留字段

## 语音播放 0x05

请求：

|  |  |
| --- | --- |
| 字段 | 索引 |
| 长度  (字节) | 1 |

响应：无

索引：给出要播放的语音的索引。语音内容是提前通过工具制作好的。

## MAC扫描 0x07

请求：无

响应：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | AP个数 | AP[0] | | | AP[N] |
| SSID | MAC | 强度 |  |
| 长度  (字节) | 1 | 16 | 6 | 2 |  |

上面是一组wifi信息，扫描时，我们会返回多组结果，根据定位需要我们最多返回前面的10组。

## 开启OTA 0x09

请求：无

响应：无

## 获取版本信息 0x0B

请求：无

响应：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 硬件版本 | 软件版本 | 编译时间 |
| 长度  (字节) | 1 | 1 | 6 |

1. socket发送基站/wifi
2. 基站/wifi响应（异步）
3. socket发送gps
4. gps响应 （异步）
5. 设置 ssid 密码
6. 设置工作模式 sta ap/加密模式
7. 请求服务器配置
8. 服务器配置响应（异步）
9. 设置socket 信息：支持多路的ip和端口设置，支持域名
10. 获取设备信息:mac，
11. 获取wifi信息： 工作模式 sta/ap，ssid| 密码
12. 播放数据,接口中是语音数据