Smart Tracker Data Parse

V1.6

Release: 2024/02/28

STATEMENT

RUICTEC Information Technology Co., LTD

Catalog

1 General Introduction 4
2 Script 5
2.1 Decode 5
2.1.1 Chirpstack 5
2.1.2 TTN 18
2.2 Encode 30
2.2.1 Chirpstack 30
2.2.2 TTN 36
3 Message in JSON 44
3.1 Uplink Message 44
3.1.1 Heartbeat 44
3.1.2 GNSS Coordinate 45
3.1.3 BLE Coordinate 46
3.1.4 Alarm 46
3.1.5 Ack 47
3.1.6 Positioning UUID List 47
3.1.7 Asset UUID List 47
3.1.8 Pass-through Filter List 48
3.1.9 History Beacon Config List 49
3.1.10 History Beacon Info List 49
3.1.11 History GNSS Info List 50
3.1.12 Pass-through Data List 51
3.2 Downlink Message 51
3.2.1 Parameters Setting 51
3.2.1.1 Tx Power 51
3.2.1.2 Data Rate 51
3.2.1.3 AUREPORT 51
3.2.1.4 BLE 52
3.2.1.5 SCAN 52
3.2.1.6 SCALE 52
3.2.1.7 BLEOFF 52
3.2.1.8 STEPSOFF 52
3.2.1.9 BUZZER 53
3.2.1.10 VIBRATOR 53
3.2.1.11 DISTANCE 53
3.2.1.12 PROXIMITY 53
3.2.1.13 GNSS Period 53
3.2.1.14 HEARTBEAT 53
3.2.1.15 DATETIME 54
3.2.1.16 SLEEPY 54
3.2.1.17 THRES 54
3.2.1.18 BLEACK 54

3.2.1.19 Multiple parameters 54
3.2.2 BLE Start Time 55
3.2.3 Command 55
3.2.4 Ack 56
3.2.5 Configure locator beacon UUID 56
3.2.6 Configure asset beacon UUID 56
3.2.7 Configure Pass-through BLE Filter 57
3.2.8 Configure Confirm needed Beacon 57

1 General Introduction

This document describes how to decode the binary data described in the datasheet into JSON format, and how to encode the JSON format into binary data, so that users can directly integrate hardware devices into their own systems. JS scripts run mainly in Lora network server, if the network server support script, the script can be configured in the server, so that the server output JSON format of data directly, it's convenient for data process of the application server. The application server can also send the parameters of the JSON format or command to the Lora network server directly. If the network server does not support script parsing, you can refer to the JS code to implement data codec on your own application server.
The script supports TTN and Chirpstack(V3/V4) network servers, and can be adjusted if you are using other servers.

This document applies for BC02,TC02,TD02,GO02,GC02.

2 Script

2.1 Decode

2.1.1 Chirpstack

Input parameters:
FPort: indicates the lora port number
Bytes: binary array. Raw data received from the base station. If it is Base64 encoded, it must be decoded first.
Chirpstack V3: https://www.rctiot.com/rct/decoder_chirpstackV3.txt
Chirpstack V4: https://www.rctiot.com/rct/decoder_chirpstackV4.txt

```
function Decode(fPort, bytes, variables) {
    var obj = {};
    var offset = 0;
    if(fPort < 10 || fPort > 27){
        obj.msgType = "unknown";
        obj.code = 1;
        obj.error = "Unknown fPort,expect([10,27]),actual(" + fPort + ")";
        return obj;
    }

    if(null == bytes){
        return {};
    }

    var indexI, indexJ;
    switch(fPort){
        //It's a heartbeat message.
    case 10:
        {
            obj.msgType = "heartbeat";
            if(bytes.length < 15){
                obj.code = 2;
                obj.error = "Wrong message length,expect(15,18,19),actual(" + bytes.length + ")";
                return obj;
            }
            obj.code = 0;

            obj.version = {};
            fwVer = bytes[0]&0x3f;
```

```
obj.version.swVer = ((bytes[0] >> 4) & 0x3).toString(16) + "." + (bytes[0]    & 0xf).toString(16);
obj.version.hardwareType = (bytes[0] >> 6);// 2:  “Vehicle tracker”, 1: "gateway" , 0: "badge";
obj.signal = {};
  if (bytes[1] <= 127)
        obj.signal.rssi = bytes[1] - 20;
else
        obj.signal.rssi = bytes[1] - 275;
if(bytes[2] <= 127)
      obj.signal.snr = bytes[2];
else
    obj.signal.snr = bytes[2]- 255;
obj.status = {};
//GNSS status
obj.status.gnss = (bytes[3] >>> 5) & 0x7; //0:off,1:positioning,2:successful,3:failed,4:indoor
obj.status.battery = (bytes[3] >> 3) & 0x3; //0:not charging,1:charging,2:complete,3:unknown
//During the heartbeat period, whether the device is moved.
obj.status.vibstate = (bytes[3] >> 2) & 0x1;//1: moved,0:static;
/*
  Only used when the hardware type is badge.
  0 indicates it works as a tracker
  1 indicates it works as a BLE gateway
*/
obj.workmode = (bytes[3] >> 1) & 0x1;//1:gateway,0:tracker;
//Calculate the available power of the battery.
var voltage = bytes[4]/ 100.0 + 2;
if(obj.version.hardwareType == 0)
{
if(voltage < 3.3){
      obj.status.soc = 0;
}
else if(voltage >= 4.15){
      obj.status.soc = 100;
}
else if(voltage >= 4.1){
      obj.status.soc = 99;
}
else if(voltage >= 4.04){
          obj.status.soc = Math.round(96 + 3 * (voltage - 4.04) / (4.1 - 4.04));
}
else if(voltage >= 3.97){
          obj.status.soc = Math.round(90 + 6 * (voltage - 3.97) / (4.04 - 3.97));
}
else if(voltage >= 3.87){
          obj.status.soc = Math.round(69 + 21 * (voltage - 3.87) / (3.97 - 3.87));
```

```
}
else if(voltage >= 3.69){
        obj.status.soc =Math.round (25 + 44 * (voltage - 3.69) / (3.87 - 3.69));
}
else if(voltage >= 3.4){
        obj.status.soc = Math.round(3 + 22 * (voltage - 3.4) / (3.69 - 3.4));
}
else if(voltage >= 3.3){
        obj.status.soc =    Math.round(3 * (voltage - 3.3) / (3.4 - 3.3));
}
}
else{
if(voltage < 3.35){
    obj.status.soc = 0;
}
else if(voltage >= 4.1){
    obj.status.soc = 100;
}
else if(voltage >= 3.6){
        obj.status.soc = Math.round(50 + 50 * (voltage - 3.6) / (4.1 - 3.6));
}
else if(voltage >= 3.35){
        obj.status.soc = Math.round( 50 * (voltage - 3.35) / (3.6 - 3.35));
}
}
//Parse the parameters.
obj.parameters = {};
obj.parameters.txPower = bytes[5] >>> 6;
//Data rate
obj.parameters.dr = (bytes[5] >> 3) & 0x7;
//Scheme
// 0:US915,1:EU868,2:AU915,3:CN470,4:AS923,5:KR920,6:IN865,7:RU864;
obj.parameters.scheme = bytes[5] & 0x7;
obj.parameters.ble = {};
obj.parameters.ble.auReport = bytes[6] >>> 7;//1:enable,0:disable;
var bleperiod = [0,5,10,20,30,60,120,300,600,900,1200,1800,3600,7200,21600,43200];
obj.parameters.ble.period = bleperiod[(bytes[6] >> 3) & 0x0f];
var scan = [1,2,3,6,9,12,15,255];
obj.parameters.ble.scan = scan[bytes[6] & 0x7];
obj.parameters.ble.scale = (bytes[7] >> 6) & 0x3;
obj.parameters.ble.stepsOff = ((bytes[7] >> 3) & 0x7) * 5;
obj.parameters.ble.bleOff = bytes[7] & 0x7;
obj.parameters.warning = {};
//buzzer 0:disable,1:enable;
```

```
obj.parameters.warning.buzzer = (bytes[8] >> 5) & 0x1;
obj.parameters.warning.vibrator = (bytes[8] >> 4) & 0x1;//1:enabled,0:disabled;
var distance = [2,4,6,8,10,15,255];
obj.parameters.warning.distance = distance[(bytes[8] >> 1) & 0x7];
obj.parameters.warning.proximity = bytes[8] & 0x1;//1:enabled,0:disabled;

//Gnss report period
var gnssperiod = [0,5,10,15,30,60,150,300,900,1800,3600,5400,10800,21600];
obj.parameters.gnssPeriod = gnssperiod[(bytes[9] >> 4) & 0xf];
//Heartbeat report period
var heartbeatperiod =
[60,300,600,1200,1800,3600,7200,21600,43200,86400,86400,86400,86400,86400,86400];
obj.parameters.heartBeatPeriod = heartbeatperiod[bytes[9] & 0xf];

if(fwVer > 0x18){
    obj.parameters.sleepy = {};
    obj.parameters.sleepy.start =    ((bytes[10] & 0x3) << 3) | ((bytes[11] >> 5) & 0x7);
    obj.parameters.sleepy.end = bytes[11] & 0x1f;
    obj.parameters.sleepy.degree = (bytes[10] >> 2) & 0x7;

  if(obj.parameters.sleepy.start != obj.parameters.sleepy.end){
    obj.parameters.timestamp = ((bytes[12] & 0xff) << 24) | ((bytes[13] & 0xff) << 16) | ((bytes[14] & 0xff) << 8)
| (bytes[15] & 0xff);
    obj.rmc = (bytes[16] >> 4) & 0x0F;
    obj.bleAck = (bytes[16] >> 3) & 0x01;
    obj.thres =    bytes[16] & 0x07;
    obj.steps = ((bytes[17] & 0xff) << 8) | (bytes[18] & 0xff);
    if(fwVer > 0x19){
      obj.temp = bytes[19] - 50;//Temperature
    }
  }
  else{
    obj.rmc = (bytes[12] >> 4) & 0x0F;
    obj.bleAck = (bytes[12] >> 3) & 0x01;
    obj.thres =    bytes[12] & 0x07;
    obj.steps = ((bytes[13] & 0xff) << 8) | (bytes[14] & 0xff);
    if(fwVer > 0x19){
      obj.temp = bytes[15] - 50;//Temperature
    }
  }
}
else{
    obj.parameters.sleepy = {};
    obj.parameters.sleepy.start =    ((bytes[10] & 0x3) << 3) | ((bytes[11] >> 5) & 0x7);
```

```
                obj.parameters.sleepy.end = bytes[11] & 0x1f;
                obj.parameters.sleepy.degree = (bytes[10] >> 2) & 0x7;

              if(obj.parameters.sleepy.start != obj.parameters.sleepy.end){
                obj.parameters.timestamp = ((bytes[12] & 0xff) << 24) | ((bytes[13] & 0xff) << 16) | ((bytes[14] & 0xff) << 8)
| (bytes[15] & 0xff);
                obj.rmc = (bytes[16] >> 4) & 0x0F;
                obj.bleAck = (bytes[16] >> 3) & 0x01;
                obj.thres =    bytes[16] & 0x07;
                obj.steps = ((bytes[18] & 0xff) << 8) | (bytes[19] & 0xff);
              }
              else{
                obj.rmc = (bytes[12] >> 4) & 0x0F;
                obj.bleAck = (bytes[12] >> 3) & 0x01;
                obj.thres =    bytes[12] & 0x07;
                obj.steps = ((bytes[14] & 0xff) << 8) | (bytes[15] & 0xff);
              }
            }
            return obj;
          }
        case 11:
          {
            obj.msgType = "GNSS coordinate";
            if(bytes.length != 9 && bytes.length != 13){
              obj.code = 2;
              obj.error = "Wrong message length,expect(9,13),actual(" + bytes.length + ")";
              return obj;
            }
            obj.code = 0;
            if(bytes.length == 9)
            {
                obj.longitude = {};
                obj.longitude.orientation = bytes[0] & 0x80    ? "W" : "E";
                var longi = (bytes[0] & 0x7f) << 24 | bytes[1] << 16 | bytes[2] << 8 | bytes[3];
                obj.longitude.value = (parseInt(longi / 10000000) + (longi % 10000000) / 6000000.0).toFixed(7);
                obj.latitude = {};
                obj.latitude.orientation = bytes[4] & 0x80    ? "S" : "N";
                var lati = (bytes[4] & 0x7f) << 24 | bytes[5] << 16 | bytes[6] << 8 | bytes[7];
                obj.latitude.value = (parseInt(lati / 10000000) + (lati % 10000000) / 6000000.0).toFixed(7);
                obj.time = bytes[8];
            }
            else if(bytes.length == 13)
            {
                obj.longitude = {};
```

```
            obj.longitude.orientation = bytes[0] & 0x80    ? "W" : "E";
            var longi = (bytes[0] & 0x7f) << 24 | bytes[1] << 16 | bytes[2] << 8 | bytes[3];
            obj.longitude.value = (parseInt(longi / 10000000) + (longi % 10000000) / 6000000.0).toFixed(7);
            obj.latitude = {};
            obj.latitude.orientation = bytes[4] & 0x80    ? "S" : "N";
            var lati = (bytes[4] & 0x7f) << 24 | bytes[5] << 16 | bytes[6] << 8 | bytes[7];
            obj.latitude.value = (parseInt(lati / 10000000) + (lati % 10000000) / 6000000.0).toFixed(7);
            var alti =    (bytes[8] & 0x7f) << 24 | bytes[8] << 16 | bytes[10] << 8 | bytes[11];
            obj.altitude = alti / 100.0;
            obj.time = bytes[12];
        }
        return obj;
    }
case 12:
    {
        obj.msgType = "BLE coordinate";
        if(bytes.length < 7){
            obj.code = 2;
            obj.error = "wrong message length,less than minimum length(10)";
            return obj;
        }
        obj.code = 0;

        var majorLen,majorShift,beaconStart;
        if(bytes[1] == 0)
        {
                obj.step = bytes[0] << 16 | bytes[1] << 8 | bytes[2];
              //How many kinds of Major value of the beacons.
                majorLen = bytes[3] & 0xf;
                majorShift = 4 + majorLen;
                beaconStart = 4;
        }
        else{
                majorLen = bytes[0] & 0xf;
                majorShift = 1 + majorLen;
                beaconStart = 1;
                obj.closecontact = (bytes[0] >> 4) & 0x1;
        }

        obj.beaconList = [];


        for(indexI = 0; indexI < majorLen; indexI++){
            var beaconNum = bytes[beaconStart + indexI];
```

```
            var major = (bytes[majorShift]    << 8 | bytes[majorShift + 1]).toString(16);

            var minorShift = majorShift + 2;
            for(indexJ = 0; indexJ < beaconNum; indexJ++){
              //var minorShift = majorShift + 2 + indexJ *3;
              var beaconObj = {};
              beaconObj.major = major;
              beaconObj.minor = ((bytes[minorShift] << 8) | bytes[minorShift + 1]).toString(16);
              beaconObj.type = (bytes[minorShift + 2] >> 5) & 0x3;//0:locator,1:asset,2:alarm,3:proximity
              beaconObj.rssi = -(bytes[minorShift + 2] & 0x1f) - 59;
              minorShift += 2;
              if(bytes[minorShift] >> 7){
                beaconObj.battery = bytes[minorShift + 1] & 0x7f;
                minorShift += 1;
              }
              minorShift += 1;
              obj.beaconList.push(beaconObj);
            }
            majorShift = minorShift;
          }
          return obj;
        }
    case 13:
        {
          obj.msgType = "alarm";
          if(bytes.length != 2){
            obj.code = 2;
            obj.error = "Wrong message length,expect(2),actual(" + bytes.length + ")";
            return obj;
          }

          obj.code = 0;
          obj.msgId = bytes[0];
          obj.ack = (bytes[1] >> 4) & 0x1;// 1:true,0:false;
      //0:SOS,1:SOS dismissed,2:power off,3:BLE disable,4:LoRa disable,5:GPS disable,6:Enter hazardous area,7:Unknown
          obj.alarm = bytes[1] & 0xf;

          return obj;
        }
    case 14:
        {
          obj.msgType = "ack";
          if(bytes.length != 2){
            obj.code = 2;
```

```
            obj.error = "Wrong message length,expect(2),actual(" + bytes.length + ")";
            return obj;
        }

        obj.code = 0;
        obj.msgId = bytes[0];
        //0:succeed,1:parameter not supported,2:parameter out of range,3:unknown;
        obj.result = bytes[1] & 0x3;

        return obj;
      }
    case 15:
      {
        obj.msgType = "Positioning UUID list";
        obj.code = 0;
        if(bytes[0] == 0){
            return {};
        }
        obj.uuidList = [];
        for(indexI = 0; indexI <    bytes[0]; indexI++){
            var uuid = {};
            uuid.index = bytes[1 + 17 * indexI];
            var str = "";
            for(indexJ = 2 + 17 * indexI; indexJ < 18 + 17 * indexI; indexJ++){
                var tmp = bytes[indexJ].toString(16);
                if(tmp.length == 1){
                    tmp = "0" + tmp;
                }
                str += tmp;
            }
            uuid.uuid = str;
            obj.uuidList.push(uuid);
        }
        return obj;
      }
    case 16:
      {
        obj.msgType = "Asset UUID list";
        obj.code = 0;
        if(bytes[0] == 0){
            return {};
        }
        obj.uuidList = [];
        for(indexI = 0; indexI <    bytes[0]; indexI++){
```

```
            var uuidAsset = {};
            uuidAsset.index = bytes[1 + 17 * indexI];
            var strAsset = "";
            for(indexJ = 2 + 17 * indexI; indexJ < 18 + 17 * indexI; indexJ++){
                var tmpAsset = bytes[indexJ].toString(16);
                if(tmpAsset.length == 1){
                    tmpAsset = "0" + tmpAsset;
                }
                strAsset += tmpAsset;
            }
            uuidAsset.uuid = strAsset;
            obj.uuidList.push(uuidAsset);
        }
        return obj;
    }
case 17:
    {
        obj.msgType = "Pass-through filter list";
        obj.code = 0;
        if(bytes[0] == 0){
            return obj;
        }
        obj.filterList = [];
        var pos = 1;
        for(indexI = 0; indexI <    bytes[0]; indexI++){
            var filterPosPass = {};
            filterPosPass.port = bytes[pos];
            filterPosPass.start = bytes[pos+1];
            filterPosPass.end = bytes[pos+2];
            filterPosPass.filterStart = bytes[pos + 3];
            filterPosPass.filterLen= bytes[pos+4];
            var filterLen = bytes[pos+4];
            var strPosPass = "";
            for(indexJ = 0; indexJ < filterLen; indexJ++){
                var tmpPosPass = bytes[pos+5+indexJ].toString(16);
                if(tmpPosPass.length == 1){
                    tmpPosPass = "0" + tmpPosPass;
                }
                strPosPass += tmpPosPass;
            }
            pos = pos + 5 + filterLen;
            filterPosPass.filter = strPosPass;
            obj.filterList.push(filterPosPass);
        }
```

```
        return obj;
    }
case 18:
{
        obj.msgType = "History Beacon Config List";
         obj.code = 0;
        if(bytes[0] == 0){
            return {};
        }
        obj.number = bytes[0];
        obj.beaconList = [];
        for(indexI = 0; indexI <    bytes[0]; indexI++){
            var beacon= {};
            offset = 5 * indexI;
            beacon.index = bytes[1 + offset];
            beacon.major =    (bytes[2 + offset]    << 8 | bytes[3 + offset]).toString(16);
            beacon.minor =    (bytes[4 + offset]    << 8 | bytes[5 + offset]).toString(16);
            obj.beaconList.push(beacon);
        }
        return obj;
}
case 19:
{
        obj.msgType = "History Beacon Info List";
         obj.code = 0;
        if(bytes[0] == 0){
            return {};
        }
        obj.number = bytes[0];
        obj.beaconList = [];
        for(indexI = 0; indexI <    bytes[0]; indexI++){
            var bea= {};
            offset = 9 * indexI;
            bea.major =    (bytes[1 + offset]    << 8 | bytes[2 + offset]).toString(16);
            bea.minor =    (bytes[3 + offset]    << 8 | bytes[4 + offset]).toString(16);
            bea.rssi = -((bytes[5 + offset] & 0x1f) + 59);
            bea.frmOff =    (bytes[6 + offset]    << 8 | bytes[7 + offset]);
            bea.timeOff =    (bytes[8 + offset]    << 8 | bytes[9 + offset]);
            obj.beaconList.push(bea);
        }
        return obj;
}
case 20:
{
```

```
        obj.msgType = "History GNSS Info List";
         obj.code = 0;
        if(bytes[0] == 0){
           return {};
        }
        obj.number = bytes[0];
        obj.gnssList = [];
        for(indexI = 0; indexI <    bytes[0]; indexI++){
           var gnss = {};
           offset = 12 * indexI;
           gnss.longitude = {};
           gnss.longitude.orientation = bytes[1+offset] & 0x80    ? "W" : "E";
        var longiH = (bytes[1+offset] & 0x7f) << 24 | bytes[2+offset] << 16 | bytes[3+offset] << 8 | bytes[4+offset];
        gnss.longitude.value = (parseInt(longiH / 10000000) + (longiH % 10000000) / 6000000.0).toFixed(7);
        gnss.latitude = {};
        gnss.latitude.orientation = bytes[5+offset] & 0x80    ? "S" : "N";
        var latiH = (bytes[5+offset] & 0x7f) << 24 | bytes[6+offset] << 16 | bytes[7+offset] << 8 | bytes[8+offset];
        gnss.latitude.value = (parseInt(latiH / 10000000) + (latiH % 10000000) / 6000000.0).toFixed(7);
        gnss.frmOff = (bytes[9 + offset]    << 8 | bytes[10 + offset]);
        gnss.timeoff =    (bytes[11 + offset]    << 8 | bytes[12 + offset]);
           obj.gnssList.push(gnss);
        }
        return obj;
    }
    case 21:
{   //Only used for RCT platform, should be removed if not needed.
        //It's an example for how to support third part BLE devices.
        obj.msgType = "Pass-through data list";
      obj.subType = "BLE Sensor";
        obj.port = fPort;
        obj.code = 0;
        if(bytes[0] == 0){
           return {};
        }
        obj.dataList = [];
        var macList = new Array();
        var dataLen = (bytes.length - 1)/bytes[0];
        for(indexI = 0; indexI <    bytes[0]; indexI++){
           var passData = {};
           passData.index =    indexI;
           var strDataPass = "";
           for(indexJ = 1 + dataLen * indexI; indexJ < 1 + dataLen + dataLen * indexI; indexJ++){
              var tmpDataPass = bytes[indexJ].toString(16);
              if(tmpDataPass.length == 1){
```

```
            tmpDataPass = "0" + tmpDataPass;
        }
        strDataPass += tmpDataPass;
    }
    passData.payload = strDataPass;

if(bytes[1 + dataLen * indexI] < 128)
    passData.x = bytes[1 + dataLen * indexI] + bytes[2 + dataLen * indexI] / 256.0;
else
    passData.x = bytes[1 + dataLen * indexI] - 256 + bytes[2 + dataLen * indexI] / 256.0;

if(bytes[3 + dataLen * indexI] < 128)
    passData.y = bytes[3 + dataLen * indexI] + bytes[4 + dataLen * indexI] / 256.0;
else
    passData.y = bytes[3 + dataLen * indexI] - 256 + bytes[4 + dataLen * indexI] / 256.0;

if(bytes[5 + dataLen * indexI] < 128)
    passData.z = bytes[5 + dataLen * indexI] + bytes[6 + dataLen * indexI] / 256.0;
else
    passData.z = bytes[5 + dataLen * indexI] - 256 + bytes[6 + dataLen * indexI] / 256.0;

passData.mac = "";
    for(indexJ = 12 + dataLen * indexI; indexJ > 6 + dataLen * indexI; indexJ--){
        var tmpDataPass = bytes[indexJ].toString(16);
        if(tmpDataPass.length == 1){
            tmpDataPass = "0" + tmpDataPass;
        }
        passData.mac += tmpDataPass;
    }
    var macExist = 0;
    for(var macIndex=0; macIndex<macList.length; macIndex++)
    {
        if(passData.mac == macList[macIndex])
        {
            macExist = 1;
            break;
        }
    }
    if(macExist == 1)
    {
        break;
    }
    else{
        macList.push(passData.mac);
```

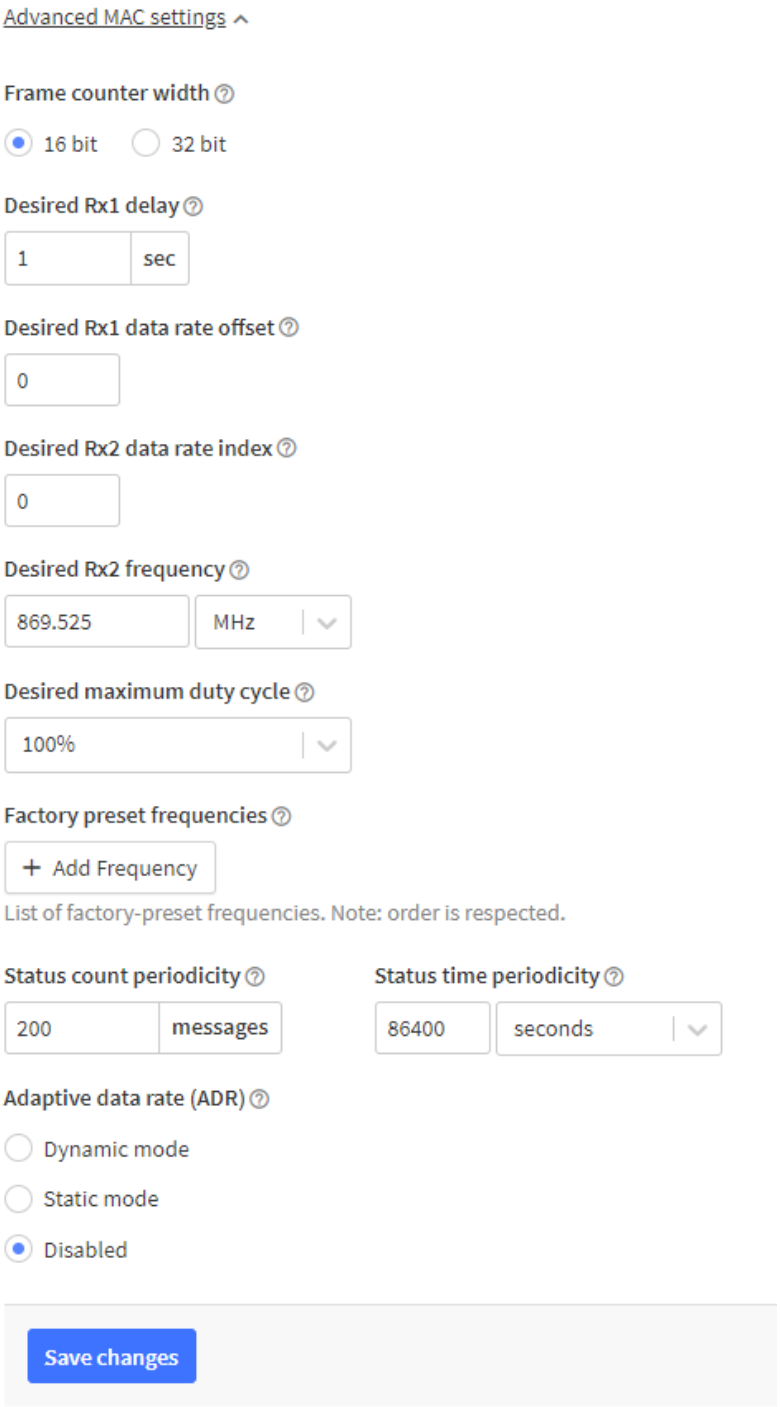
```
                }
                passData.rssi = bytes[13 + dataLen * indexI] - 255;
                if(passData.y < -0.8 && Math.abs(passData.x) <= 0.1 && Math.abs(passData.z) < 0.1)
               passData.state = 0;//正常
            else if(Math.abs(passData.y) < 0.1){
                        passData.state = 2;
                }
                else
               passData.state = 1;//倾斜

                obj.dataList.push(passData);
            }
            return obj;
        }
    case 22:
    case 23:
    case 24:
    case 25:
        {
            obj.msgType = "Pass-through data list";
            obj.port = fPort;
            obj.code = 0;
            if(bytes[0] == 0){
                return {};
            }
            obj.dataList = [];
            var dataLen = (bytes.length - 1)/bytes[0];
            for(indexI = 0; indexI <    bytes[0]; indexI++){
                var passData = {};
                passData.index =    indexI;
                var strDataPass = "";
                for(indexJ = 1 + dataLen * indexI; indexJ < 1 + dataLen + dataLen * indexI; indexJ++){
                    var tmpDataPass = bytes[indexJ].toString(16);
                    if(tmpDataPass.length == 1){
                        tmpDataPass = "0" + tmpDataPass;
                    }
                    strDataPass += tmpDataPass;
                }
                passData.payload = strDataPass;
                obj.dataList.push(passData);
            }
            return obj;
        }
    default:
```

```
            return {};
        }
        return {};
    }
```

2.1.2 TTN

Download: https://www.rctiot.com/rct/decoder_TTN.txt

In TTN, the device should follow the configuration below:



```
function decodeUplink(input) {
    fPort = input.fPort;
    bytes = input.bytes;
    var obj = Decode(fPort, bytes);
    return {
        data: obj,
        warnings: [],
        errors: []
    };
}
function Decode(fPort, bytes) {
    var obj = {};
    var offset = 0;
    if( fPort < 10 || fPort > 25){
        obj.msgType = "unknown";
        obj.code = 1;
        obj.error = "Unknown fPort,expect([10,25]),actual(" + fPort + ")";
        return obj;
    }

    if(null === bytes){
        return {};
    }

    var indexI, indexJ;
    switch(fPort){
          //It's a heartbeat message.
        case 10:
            {
                obj.msgType = "heartbeat";
                if(bytes.length < 15){
                    obj.code = 2;
                    obj.error = "Wrong message length,expect(15,18,19),actual(" + bytes.length + ")";
                    return obj;
                }
                obj.code = 0;

                obj.version = {};
                fwVer = bytes[0]&0x3f;
                obj.version.swVer = ((bytes[0] >> 4) & 0x3).toString() + "." + (bytes[0]    & 0xf).toString();
                obj.version.hardwareType = (bytes[0] >> 6);// 1: "gateway" , 0: "badge";
                obj.signal = {};
                 if (bytes[1] <= 127)
                      obj.signal.rssi = bytes[1] - 20;
```

```
    else
        obj.signal.rssi = bytes[1] - 275;
    if(bytes[2] <= 127)
        obj.signal.snr = bytes[2];
    else
       obj.signal.snr = bytes[2]- 255;
    obj.status = {};
    //GNSS status
    obj.status.gnss = (bytes[3] >>> 5) & 0x7; //0:off,1:positioning,2:successful,3:failed,4:Indoor,5:Stationary
    obj.status.battery = (bytes[3] >> 3) & 0x3; //0:not charging,1:charging,2:complete,3:unknown
    //During the heartbeat period, whether the device is moved.
    obj.status.vibstate = (bytes[3] >> 2) & 0x1;//1: moved,0:static;
    /*
      Only used when the hardware type is badge.
      0 indicates it works as a tracker
      1 indicates it works as a BLE gateway
    */
    obj.workmode = (bytes[3] >> 1) & 0x1;//1:gateway,0:tracker;
    //Calculate the available power of the battery.
    var voltage = bytes[4]/ 100.0 + 2;
if(obj.version.hardwareType == 0)
{
    if(voltage < 3.3){
        obj.status.soc = 0;
    }
    else if(voltage >= 4.15){
        obj.status.soc = 100;
    }
    else if(voltage >= 4.1){
        obj.status.soc = 99;
  }
  else if(voltage >= 4.04){
          obj.status.soc = Math.round(96 + 3 * (voltage - 4.04) / (4.1 - 4.04));
    }
    else if(voltage >= 3.97){
          obj.status.soc = Math.round(90 + 6 * (voltage - 3.97) / (4.04 - 3.97));
    }
    else if(voltage >= 3.87){
          obj.status.soc = Math.round(69 + 21 * (voltage - 3.87) / (3.97 - 3.87));
    }
    else if(voltage >= 3.69){
          obj.status.soc =Math.round (25 + 44 * (voltage - 3.69) / (3.92 - 3.69));
    }
    else if(voltage >= 3.4){
```

```
        obj.status.soc = Math.round(3 + 22 * (voltage - 3.4) / (3.69 - 3.4));
 }
 else if(voltage >= 3.3){
        obj.status.soc =    Math.round(3 * (voltage - 3.3) / (3.4 - 3.3));
 }
 }
Else{
 if(voltage < 3.35){
     obj.status.soc = 0;
 }
 else if(voltage >= 4.1){
     obj.status.soc = 100;
 }
else if(voltage >= 3.6){
        obj.status.soc = Math.round(50 + 50 * (voltage - 3.6) / (4.1 - 3.6));
 }
 else if(voltage >= 3.35){
        obj.status.soc = Math.round( 50 * (voltage - 3.35) / (3.6 - 3.35));
 }
 }
 //Parse the parameters.
 obj.parameters = {};
 obj.parameters.txPower = bytes[5] >>> 6;
 //Data rate
 obj.parameters.dr = (bytes[5] >> 3) & 0x7;
 //Scheme
 // 0:US915,1:EU868,2:AU915,3:CN470,4:AS923,5:KR920,6:IN865,7:RU864;
 obj.parameters.scheme = bytes[5] & 0x7;
 obj.parameters.ble = {};
 obj.parameters.ble.auReport = bytes[6] >>> 7;//1:enable,0:disable;
 var bleperiod = [0,5,10,20,30,60,120,300,600,900,1200,1800,3600,7200,21600,43200];
 obj.parameters.ble.period = bleperiod[(bytes[6] >> 3) & 0x0f];
 var scan = [1,2,3,6,9,12,15,255];
 obj.parameters.ble.scan = scan[bytes[6] & 0x7];
 obj.parameters.ble.scale = (bytes[7] >> 6) & 0x3;
 obj.parameters.ble.stepsOff = ((bytes[7] >> 3) & 0x7) * 2;
 obj.parameters.ble.bleOff = bytes[7] & 0x7;
 obj.parameters.warning = {};
 //buzzer 0:disable,1:enable;
 obj.parameters.warning.buzzer = (bytes[8] >> 5) & 0x1;
 obj.parameters.warning.vibrator = (bytes[8] >> 4) & 0x1;//1:enabled,0:disabled;
 var distance = [2,4,6,8,10,15,255];
 obj.parameters.warning.distance = distance[(bytes[8] >> 1) & 0x7];
 obj.parameters.warning.proximity = bytes[8] & 0x1;//1:enabled,0:disabled;
```

```
        //Gnss report period
        var gnssperiod = [0,5,10,15,30,60,150,300,900,1800,3600,5400,10800,21600];
        obj.parameters.gnssPeriod = gnssperiod[(bytes[9] >> 4) & 0xf];
        //Heartbeat report period
        var heartbeatperiod =
[60,300,600,1200,1800,3600,7200,21600,43200,86400,86400,86400,86400,86400,86400,86400];
        obj.parameters.heartBeatPeriod = heartbeatperiod[bytes[9] & 0xf];

        if(fwVer > 0x18){
            obj.parameters.sleepy = {};
            obj.parameters.sleepy.start =    ((bytes[10] & 0x3) << 3) | ((bytes[11] >> 5) & 0x7);
            obj.parameters.sleepy.end = bytes[11] & 0x1f;
            obj.parameters.sleepy.degree = (bytes[10] >> 2) & 0x7;

          if(obj.parameters.sleepy.start != obj.parameters.sleepy.end){
            obj.parameters.timestamp = ((bytes[12] & 0xff) << 24) | ((bytes[13] & 0xff) << 16) | ((bytes[14] & 0xff) << 8)
| (bytes[15] & 0xff);
            obj.rmc = (bytes[16] >> 4) & 0x0F;
            obj.bleAck = (bytes[16] >> 3) & 0x01;
            obj.thres =    bytes[16] & 0x07;
            obj.steps = ((bytes[17] & 0xff) << 8) | (bytes[18] & 0xff);
            if(fwVer > 0x19){
              obj.temp = bytes[19] - 50;//Temperature
            }
          }
          else{
            obj.rmc = (bytes[12] >> 4) & 0x0F;
            obj.bleAck = (bytes[12] >> 3) & 0x01;
            obj.thres =    bytes[12] & 0x07;
            obj.steps = ((bytes[13] & 0xff) << 8) | (bytes[14] & 0xff);
            if(fwVer > 0x19){
              obj.temp = bytes[15] - 50;//Temperature
            }
          }
        }
        else{
            obj.parameters.sleepy = {};
            obj.parameters.sleepy.start =    ((bytes[10] & 0x3) << 3) | ((bytes[11] >> 5) & 0x7);
            obj.parameters.sleepy.end = bytes[11] & 0x1f;
            obj.parameters.sleepy.degree = (bytes[10] >> 2) & 0x7;

          if(obj.parameters.sleepy.start != obj.parameters.sleepy.end){
            obj.parameters.timestamp = ((bytes[12] & 0xff) << 24) | ((bytes[13] & 0xff) << 16) | ((bytes[14] & 0xff) << 8)
```

```
| (bytes[15] & 0xff);
                    obj.rmc = (bytes[16] >> 4) & 0x0F;
                    obj.bleAck = (bytes[16] >> 3) & 0x01;
                    obj.thres =    bytes[16] & 0x07;
                    obj.steps = ((bytes[18] & 0xff) << 8) | (bytes[19] & 0xff);
                }
                else{
                    obj.rmc = (bytes[12] >> 4) & 0x0F;
                    obj.bleAck = (bytes[12] >> 3) & 0x01;
                    obj.thres =    bytes[12] & 0x07;
                    obj.steps = ((bytes[14] & 0xff) << 8) | (bytes[15] & 0xff);
                }
            }
            return obj;
        }
        case 11:
        {
            obj.msgType = "GNSS coordinate";
            if(bytes.length != 9 && bytes.length != 13){
                obj.code = 2;
                obj.error = "Wrong message length,expect(9,13),actual(" + bytes.length + ")";
                return obj;
            }
            obj.code = 0;
            if(bytes.length == 9)
            {
                obj.longitude = {};
                obj.longitude.orientation = bytes[0] & 0x80    ? "W" : "E";
                var longi = (bytes[0] & 0x7f) << 24 | bytes[1] << 16 | bytes[2] << 8 | bytes[3];
                obj.longitude.value = (parseInt(longi / 10000000) + (longi % 10000000) / 6000000.0).toFixed(7);
                obj.latitude = {};
                obj.latitude.orientation = bytes[4] & 0x80    ? "S" : "N";
                var lati = (bytes[4] & 0x7f) << 24 | bytes[5] << 16 | bytes[6] << 8 | bytes[7];
                obj.latitude.value = (parseInt(lati / 10000000) + (lati % 10000000) / 6000000.0).toFixed(7);
                obj.time = bytes[8];
            }
            else if(bytes.length == 13)
            {
                obj.longitude = {};
                obj.longitude.orientation = bytes[0] & 0x80    ? "W" : "E";
                var longi = (bytes[0] & 0x7f) << 24 | bytes[1] << 16 | bytes[2] << 8 | bytes[3];
                obj.longitude.value = (parseInt(longi / 10000000) + (longi % 10000000) / 6000000.0).toFixed(7);
                obj.latitude = {};
                obj.latitude.orientation = bytes[4] & 0x80    ? "S" : "N";
```

```
            var lati = (bytes[4] & 0x7f) << 24 | bytes[5] << 16 | bytes[6] << 8 | bytes[7];
            obj.latitude.value = (parseInt(lati / 10000000) + (lati % 10000000) / 6000000.0).toFixed(7);
            var alti =    (bytes[8] & 0x7f) << 24 | bytes[8] << 16 | bytes[10] << 8 | bytes[11];
            obj.altitude = alti / 100.0;
            obj.time = bytes[12];
        }
        return obj;
}
   case 12:
     {
        obj.msgType = "BLE coordinate";
        if(bytes.length <    7){
          obj.code = 2;
          obj.error = "wrong message length,less than minimum length(10)";
          return obj;
        }
        obj.code = 0;
        var majorLen,majorShift,beaconStart;
        if(bytes[1] == 0)
        {
              obj.step = bytes[0] << 16 | bytes[1] << 8 | bytes[2];
            //How many kinds of Major value of the beacons.
              majorLen = bytes[3] & 0xf;
              majorShift = 4 + majorLen;
              beaconStart = 4;
        }
        else{
             majorLen = bytes[0] & 0xf;
             majorShift = 1 + majorLen;
             beaconStart = 1;
             obj.closecontact = (bytes[0] >> 4) & 0x1;
        }
        obj.beaconList = [];

        for(indexI = 0; indexI < majorLen; indexI++){
          var beaconNum = bytes[beaconStart + indexI];
          var major = (bytes[majorShift]    << 8 | bytes[majorShift + 1]).toString(16);

          var minorShift = majorShift + 2;
          for(indexJ = 0; indexJ < beaconNum; indexJ++){
            //var minorShift = majorShift + 2 + indexJ *3;
            var beaconObj = {};
            beaconObj.major = major;
            beaconObj.minor = ((bytes[minorShift] << 8) | bytes[minorShift + 1]).toString(16);
```

```
            beaconObj.type = (bytes[minorShift + 2] >> 5) & 0x3;//0:locator,1:asset,2:alarm,3:proximity
            beaconObj.rssi = -(bytes[minorShift + 2] & 0x1f) - 59;
            minorShift += 2;
            if(bytes[minorShift] >> 7){
              beaconObj.battery = bytes[minorShift + 1] & 0x7f;
              minorShift += 1;
            }
            minorShift += 1;
            obj.beaconList.push(beaconObj);
          }
          majorShift = minorShift;
        }
        return obj;
      }
    case 13:
      {
        obj.msgType = "alarm";
        if(bytes.length != 2){
          obj.code = 2;
          obj.error = "Wrong message length,expect(2),actual(" + bytes.length + ")";
          return obj;
        }

        obj.code = 0;
        obj.msgId = bytes[0];
        obj.ack = (bytes[1] >> 4) & 0x1;// 1:true,0:false;
      //0:SOS,1:SOS dismissed,2:power off,3:BLE disable,4:LoRa disable,5:GPS disable,6:Enter hazardous area,7:Unknown
        obj.alarm = bytes[1] & 0x7;

        return obj;
      }
    case 14:
      {
        obj.msgType = "ack";
        if(bytes.length != 2){
          obj.code = 2;
          obj.error = "Wrong message length,expect(2),actual(" + bytes.length + ")";
          return obj;
        }

        obj.code = 0;
        obj.msgId = bytes[0];
        //0:succeed,1:parameter not supported,2:parameter out of range,3:unknown;
        obj.result = bytes[1] & 0x3;
```

```
        return obj;
      }
case 15:
    {
        obj.msgType = "Positioning UUID list";
        obj.code = 0;
        if(bytes[0] == 0){
            return {};
        }
        obj.uuidList = [];
        for(indexI = 0; indexI <    bytes[0]; indexI++){
            var uuid = {};
            uuid.index = bytes[1 + 17 * indexI];
            var str = "";
            for(indexJ = 2 + 17 * indexI; indexJ < 18 + 17 * indexI; indexJ++){
                var tmp = bytes[indexJ].toString(16);
                if(tmp.length == 1){
                    tmp = "0" + tmp;
                }
                str += tmp;
            }
            uuid.uuid = str;
            obj.uuidList.push(uuid);
        }
        return obj;
    }
case 16:
    {
        obj.msgType = "Asset UUID list";
        obj.code = 0;
        if(bytes[0] == 0){
            return {};
        }
        obj.uuidList = [];
        for(indexI = 0; indexI <    bytes[0]; indexI++){
            var uuidAsset = {};
            uuidAsset.index = bytes[1 + 17 * indexI];
            var strAsset = "";
            for(indexJ = 2 + 17 * indexI; indexJ < 18 + 17 * indexI; indexJ++){
                var tmpAsset = bytes[indexJ].toString(16);
                if(tmpAsset.length == 1){
                    tmpAsset = "0" + tmpAsset;
                }
```

```
            strAsset += tmpAsset;
        }
        uuidAsset.uuid = strAsset;
        obj.uuidList.push(uuidAsset);
    }
    return obj;
  }
case 17:
  {
    obj.msgType = "Pass-through filter list";
    obj.code = 0;
    obj.length = bytes[0];
    if(bytes[0] == 0){
        return obj;
    }
    obj.filterList = [];
    var pos = 1;
    for(indexI = 0; indexI <    bytes[0]; indexI++){
        var filterPosPass = {};
        filterPosPass.port = bytes[pos];
        filterPosPass.start = bytes[pos+1];
        filterPosPass.end = bytes[pos+2];
        filterPosPass.filterStart = bytes[pos + 3];
        filterPosPass.filterLen= bytes[pos+4];
        var filterLen = bytes[pos+4];
        var strPosPass = "";
        for(indexJ = 0; indexJ < filterLen; indexJ++){
            var tmpPosPass = bytes[pos+5+indexJ].toString(16);
            if(tmpPosPass.length == 1){
                tmpPosPass = "0" + tmpPosPass;
            }
            strPosPass += tmpPosPass;
        }
        pos = pos + 5 + filterLen;
        filterPosPass.filter = strPosPass;
        obj.filterList.push(filterPosPass);
    }
    return obj;
  }
case 18:
{
    obj.msgType = "History Beacon Config List";
     obj.code = 0;
    if(bytes[0] == 0){
```

```
            return {};
        }
        obj.number = bytes[0];
        obj.beaconList = [];
        for(indexI = 0; indexI <    bytes[0]; indexI++){
            var beacon= {};
            offset = 5 * indexI;
            beacon.index = bytes[1 + offset];
            beacon.major =    (bytes[2 + offset]    << 8 | bytes[3 + offset]).toString(16);
            beacon.minor =    (bytes[4 + offset]    << 8 | bytes[5 + offset]).toString(16);
            obj.beaconList.push(beacon);
        }
        return obj;
    }
    case 19:
    {
        obj.msgType = "History Beacon Info List";
         obj.code = 0;
        if(bytes[0] == 0){
            return {};
        }
        obj.number = bytes[0];
        obj.beaconList = [];
        for(indexI = 0; indexI <    bytes[0]; indexI++){
            var bea= {};
            offset = 9 * indexI;
            bea.major =    (bytes[1 + offset]    << 8 | bytes[2 + offset]).toString(16);
            bea.minor =    (bytes[3 + offset]    << 8 | bytes[4 + offset]).toString(16);
            bea.rssi = -((bytes[5 + offset] & 0x1f) + 59);
            bea.frmOff =    (bytes[6 + offset]    << 8 | bytes[7 + offset]);
            bea.timeOff =    (bytes[8 + offset]    << 8 | bytes[9 + offset]);
            obj.beaconList.push(bea);
        }
        return obj;
    }
    case 20:
    {
        obj.msgType = "History GNSS Info List";
         obj.code = 0;
        if(bytes[0] == 0){
            return {};
        }
        obj.number = bytes[0];
        obj.gnssList = [];
```

```
        for(indexI = 0; indexI <    bytes[0]; indexI++){
            var gnss = {};
            offset = 12 * indexI;
            gnss.longitude = {};
            gnss.longitude.orientation = bytes[1+offset] & 0x80    ? "W" : "E";
        var longiH = (bytes[1+offset] & 0x7f) << 24 | bytes[2+offset] << 16 | bytes[3+offset] << 8 | bytes[4+offset];
        gnss.longitude.value = (parseInt(longiH / 10000000) + (longiH % 10000000) / 6000000.0).toFixed(7);
        gnss.latitude = {};
        gnss.latitude.orientation = bytes[5+offset] & 0x80    ? "S" : "N";
        var latiH = (bytes[5+offset] & 0x7f) << 24 | bytes[6+offset] << 16 | bytes[7+offset] << 8 | bytes[8+offset];
        gnss.latitude.value = (parseInt(latiH / 10000000) + (latiH % 10000000) / 6000000.0).toFixed(7);
        gnss.frmOff = (bytes[9 + offset]    << 8 | bytes[10 + offset]);
        gnss.timeoff =    (bytes[11 + offset]    << 8 | bytes[12 + offset]);
            obj.gnssList.push(gnss);
        }
        return obj;
    }
    case 21:
    case 22:
    case 23:
    case 24:
    case 25:
      {
        obj.msgType = "Pass-through data list";
        obj.port = fPort;
        obj.code = 0;
        if(bytes[0] == 0){
            return {};
        }
        obj.dataList = [];
        var dataLen = (bytes.length - 1)/bytes[0];
        for(indexI = 0; indexI <    bytes[0]; indexI++){
            var passData = {};
            passData.index =    indexI;
            var strDataPass = "";
            for(indexJ = 1 + dataLen * indexI; indexJ < 1 + dataLen + dataLen * indexI; indexJ++){
                var tmpDataPass = bytes[indexJ].toString(16);
                if(tmpDataPass.length == 1){
                    tmpDataPass = "0" + tmpDataPass;
                }
                strDataPass += tmpDataPass;
            }
            passData.payload = strDataPass;
            obj.dataList.push(passData);
```

```
            }
            return obj;
        }
    default:
        return {};
    }
    return {};
}
```

2.2 Encode

2.2.1 Chirpstack

Input parameters:

FPort: indicates the downlink Lora port.

Obj: parameter or command in JSON format.
Chirpstack V3: https://www.rctiot.com/rct/encoder_chirpstackV3.txt
Chirpstack V4: https://www.rctiot.com/rct/encoder_chirpstackV4.txt

```
function Encode(fPort, obj) {
    var rst = [];
    if(fPort < 10 || fPort > 17){
        console.log("Unknown fPort,expect([10,17]),actual(" + fPort + ")");
        return rst;
    }

    if(null == obj){
        console.log("obj is null");
        return [];
    }

     if(obj.msgId <0 || obj.msgId > 255){
        return [];
     }
    var arrayIndex = 0;
    switch(fPort){
        //It's a parameters setting message.
    case 10:
        {
            var index;
```

```
rst[arrayIndex++] = obj.msgId;
var power = [0,1,2,3];
index = power.indexOf(obj.txPower);
if(-1 != index){
    rst[arrayIndex++] = 1;
    rst[arrayIndex++] = index;
}

var dr = [0,1,2];
index = dr.indexOf(obj.dr);
if(-1 != index){
    rst[arrayIndex++] = 2;
    rst[arrayIndex++] = index;
}

var auReport = ["disable","enable"];
index = auReport.indexOf(obj.bleAuReport);
if(-1 != index){
    rst[arrayIndex++] = 3;
    rst[arrayIndex++] = index;
}

var blePeriod = [0,5,10,20,30,60,120,300,600,900,1200,1800,3600,7200,21600,43200];
index = blePeriod.indexOf(obj.blePeriod);
if(-1 != index){
    rst[arrayIndex++] = 4;
    rst[arrayIndex++] = index;
}

var bleScan = [1,2,3,6,9,12,15,255];
index = bleScan.indexOf(obj.bleScan);
if(-1 != index){
    rst[arrayIndex++] = 5;
    rst[arrayIndex++] = index;
}

var scale = [0,1,2,3];
index = scale.indexOf(obj.scale);
if(-1 != index){
    rst[arrayIndex++] = 6;
    rst[arrayIndex++] = index;
}

var bleStepsOff = [0,1,2,3,4,5,6,7];
```

```
index = bleStepsOff.indexOf(obj.bleStepsOff);
if(-1 != index){
    rst[arrayIndex++] = 7;
    rst[arrayIndex++] = index;
}

var bleBleOff = [0,1,2,3,4,5,6,7];
index = bleBleOff.indexOf(obj.bleBleOff);
if(-1 != index){
    rst[arrayIndex++] = 8;
    rst[arrayIndex++] = index;
}

var warnBuzzer = ["disable","enable"];
index = warnBuzzer.indexOf(obj.warnBuzzer);
if(-1 != index){
    rst[arrayIndex++] = 9;
    rst[arrayIndex++] = index;
}

var warnVibrator = ["disable","enable"];
index = warnVibrator.indexOf(obj.warnVibrator);
if(-1 != index){
    rst[arrayIndex++] = 10;
    rst[arrayIndex++] = index;
}

var warnDistance = [2,4,6,8,10,15,255];
index = warnDistance.indexOf(obj.warnDistance);
if(-1 != index){
    rst[arrayIndex++] = 11;
    rst[arrayIndex++] = index;
}

var warnProximity = ["disable","enable"];
index = warnProximity.indexOf(obj.warnProximity);
if(-1 != index){
    rst[arrayIndex++] = 12;
    rst[arrayIndex++] = index;
}

var gnssPeriod = [0,10,20,30,60,120,300,600,1800,3600,7200,10800,21600,43200];
index = gnssPeriod.indexOf(obj.gnssPeriod);
if(-1 != index){
```

```
    rst[arrayIndex++] = 13;
    rst[arrayIndex++] = index;
  }

  var heartBeatPeriod = [60,300,600,1200,1800,3600,7200,21600,43200,86400];
  index = heartBeatPeriod.indexOf(obj.heartBeatPeriod);
  if(-1 != index){
    rst[arrayIndex++] = 14;
    rst[arrayIndex++] = index;
  }

  if(null != obj.sleepy){
    if(obj.sleepy.start >= 0 && obj.sleepy.start <=23 &&
        obj.sleepy.end >= 0 && obj.sleepy.end <= 23 &&
        obj.sleepy.degree >=0 && obj.sleepy.degree <=7){
        rst[arrayIndex++] = 16;
        rst[arrayIndex++] = obj.sleepy.degree;
        rst[arrayIndex++] = obj.sleepy.start;
        rst[arrayIndex++] = obj.sleepy.end;
    }
  }

var bleThres= [0,1,2,3,4,5,6,7];
  index = bleThres.indexOf(obj.thres);
  if(-1 != index){
    rst[arrayIndex++] = 17;
    rst[arrayIndex++] = index;
  }

var bleAck = [0,1];
  index = bleAck.indexOf(obj.bleAck);
  if(-1 == index){
      var bleAckStr= ["disable","enable"];
      index = bleAckStr.indexOf(obj.bleAck);
      if(-1 != index){
            rst[arrayIndex++] = 18;
            rst[arrayIndex++] = index;
       }
  }
else{
          rst[arrayIndex++] = 18;
          rst[arrayIndex++] = index;
}
```

```
        return rst;
    }
    break;
case 11:
{
            rst[arrayIndex++] = obj.msgId;
            rst[arrayIndex++] = obj.time >> 8;
            rst[arrayIndex++] = obj.time & 0xFF;
            return rst;
}
break;
    //command
case 12:
    {
        rst[arrayIndex++] = obj.msgId;
        if(obj.cmd >= 0 && obj.cmd <= 9){
            rst[arrayIndex] = obj.cmd;
            return rst;
        }
    }
    break;
    //ack
case 13:
    {
        rst[arrayIndex] = obj.msgId;
        return rst;
    }
    break;
  //Positioning beacon UUID setting
case 14:
    //Asset beacon UUID setting
case 15:
    {
         var uuidNum = obj.uuidList.length;
         rst[arrayIndex++] = obj.msgId;
         rst[arrayIndex++] = uuidNum;
         if(uuidNum >= 1 && uuidNum <=5){
            for(var i=0; i<uuidNum; i++){
                var uuidIndex = obj.uuidList[i].index;
                if(uuidIndex >= 0 && uuidIndex <= 4){
                    var uuid = obj.uuidList[i].uuid;
                    if(uuid.length == 32){
                        rst[arrayIndex++] = uuidIndex;
                        var pos=0;
```

```
                    for(var j=0; j<16; j++)
                    {
                        var s = uuid.substr(pos, 2);
                        var v = parseInt(s, 16);
                        rst[arrayIndex++] = v;
                        pos += 2;
                    }
                }
                else{
                    return [];
                }
            }
        }
        return rst;
    }
}
break;
case 16:
{
    rst[arrayIndex++] = obj.msgId;
    var filterPort = obj.port;
    if(filterPort >= 21 && filterPort <= 25){
        var filterLen = obj.filterLen;
        var filter = obj.filter;
        if(filter.length == (filterLen<<1)){
            rst[arrayIndex++] = filterPort;
            rst[arrayIndex++] = obj.start;
            rst[arrayIndex++] = obj.end;
            rst[arrayIndex++] = obj.filterStart;
            rst[arrayIndex++] = obj.filterLen;
            var filterPos=0;
            for(var j1=0; j1<filterLen ; j1++)
            {
                var s1 = filter.substr(filterPos, 2);
                var v1 = parseInt(s1, 16);
                rst[arrayIndex++] = v1;
                filterPos += 2;
            }
        }
        return rst;
    }
}
break;
case 17:
```

```
        {
            var beaconNum = obj.number;
            rst[arrayIndex++] = obj.msgId;
            rst[arrayIndex++] = beaconNum;
            if(beaconNum >= 1 && beaconNum <=20){
              for(var beaconI=0; beaconI<beaconNum; beaconI++){
                var beaconIndex = obj.beaconList[beaconI].index;
                if(beaconIndex >= 0 && beaconIndex <= 19){
                    var beaconMajor= obj.beaconList[beaconI].major;
                    var beaconMinor= obj.beaconList[beaconI].minor;
                    rst[arrayIndex++] = beaconIndex;
                    rst[arrayIndex++] =    beaconMajor >> 8;
                    rst[arrayIndex++] = beaconMajor & 0xff;
                    rst[arrayIndex++] =    beaconMinor >> 8;
                    rst[arrayIndex++] = beaconMinor & 0xff;
                  }
              }
              return rst;
            }
          }
          break;
        default:
      }

      return [];
}
```

2.2.2 TTN

```
Download: https://www.rctiot.com/rct/encoder_TTN.txt
function encodeDownlink(input) {
  var fPort = input.fPort;
  var obj = input.data;
  var rst = [];
  fPort =Number(fPort);
  if(fPort < 10 || fPort > 17){
    return {
    bytes: [],
    fPort: fPort ,
    warnings: [],
    errors: "Unknown fPort,expect([10,17]),actual(" + fPort + ")"
    };
  }
```

```
 if(obj.msgId <0 || obj.msgId > 255){
            return {
                bytes: [],
                fPort: fPort ,
                warnings: [],
                errors: []
   };
 }
var arrayIndex = 0;
switch(fPort){
      //It's a parameters setting message.
   case 10:
      {
         var index;
         rst[arrayIndex++] = obj.msgId;
         var power = [0,1,2,3];
         index = power.indexOf(obj.txPower);
         if(-1 != index){
            rst[arrayIndex++] = 1;
            rst[arrayIndex++] = index;
         }

         var dr = [0,1,2];
         index = dr.indexOf(obj.dr);
         if(-1 != index){
            rst[arrayIndex++] = 2;
            rst[arrayIndex++] = index;
         }

         var auReport = ["disable","enable"];
         index = auReport.indexOf(obj.bleAuReport);
         if(-1 != index){
            rst[arrayIndex++] = 3;
            rst[arrayIndex++] = index;
         }

         var blePeriod = [0,5,10,20,30,60,120,300,600,900,1200,1800,3600,7200,21600,43200];
         index = blePeriod.indexOf(obj.blePeriod);
         if(-1 != index){
            rst[arrayIndex++] = 4;
            rst[arrayIndex++] = index;
         }

         var bleScan = [1,2,3,6,9,12,15,255];
```

```
index = bleScan.indexOf(obj.bleScan);
if(-1 != index){
    rst[arrayIndex++] = 5;
    rst[arrayIndex++] = index;
}

var scale = [0,1,2,3];
index = scale.indexOf(obj.scale);
if(-1 != index){
    rst[arrayIndex++] = 6;
    rst[arrayIndex++] = index;
}

var bleStepsOff = [0,1,2,3,4,5,6,7];
index = bleStepsOff.indexOf(obj.bleStepsOff);
if(-1 != index){
    rst[arrayIndex++] = 7;
    rst[arrayIndex++] = index;
}

var bleBleOff = [0,1,2,3,4,5,6,7];
index = bleBleOff.indexOf(obj.bleBleOff);
if(-1 != index){
    rst[arrayIndex++] = 8;
    rst[arrayIndex++] = index;
}

var warnBuzzer = ["disable","enable"];
index = warnBuzzer.indexOf(obj.warnBuzzer);
if(-1 != index){
    rst[arrayIndex++] = 9;
    rst[arrayIndex++] = index;
}

var warnVibrator = ["disable","enable"];
index = warnVibrator.indexOf(obj.warnVibrator);
if(-1 != index){
    rst[arrayIndex++] = 10;
    rst[arrayIndex++] = index;
}

var warnDistance = [2,4,6,8,10,15,255];
index = warnDistance.indexOf(obj.warnDistance);
if(-1 != index){
```

```
    rst[arrayIndex++] = 11;
    rst[arrayIndex++] = index;
  }

  var warnProximity = ["disable","enable"];
  index = warnProximity.indexOf(obj.warnProximity);
  if(-1 != index){
    rst[arrayIndex++] = 12;
    rst[arrayIndex++] = index;
  }
  var gnssPeriod = [0,10,20,30,60,120,300,600,1800,3600,7200,10800,21600,43200];
  index = gnssPeriod.indexOf(obj.gnssPeriod);
  if(-1 != index){
    rst[arrayIndex++] = 13;
    rst[arrayIndex++] = index;
  }
  var heartBeatPeriod = [60,300,600,1200,1800,3600,7200,21600,43200,86400];
  index = heartBeatPeriod.indexOf(obj.heartBeatPeriod);
  if(-1 != index){
    rst[arrayIndex++] = 14;
    rst[arrayIndex++] = index;
  }
  if(null != obj.sleepy){
    if(obj.sleepy.start >= 0 && obj.sleepy.start <=23 &&
        obj.sleepy.end >= 0 && obj.sleepy.end <= 23 &&
        obj.sleepy.degree >=0 && obj.sleepy.degree <=7){
        rst[arrayIndex++] = 16;
        rst[arrayIndex++] = obj.sleepy.degree;
        rst[arrayIndex++] = obj.sleepy.start;
        rst[arrayIndex++] = obj.sleepy.end;
    }
  }
 var bleThres= [0,1,2,3,4,5,6,7];
  index = bleThres.indexOf(obj.thres);
  if(-1 != index){
    rst[arrayIndex++] = 17;
    rst[arrayIndex++] = index;
  }
 var bleAck = [0,1];
  index = bleAck.indexOf(obj.bleAck);

 if(-1 == index){
      var bleAckStr= ["disable","enable"];
      index = bleAckStr.indexOf(obj.bleAck);
```

```
            if(-1 != index){
                rst[arrayIndex++] = 18;
                rst[arrayIndex++] = index;
            }
        }
        else{
            rst[arrayIndex++] = 18;
            rst[arrayIndex++] = index;
        }
        return {
            bytes: rst,
            fPort: fPort ,
            warnings: [],
            errors: []
            };
    }
    break;
case 11:
{
            rst[arrayIndex++] = obj.msgId;
            rst[arrayIndex++] = obj.time >> 8;
            rst[arrayIndex++] = obj.time & 0xFF;
            return {
            bytes: rst,
            fPort: fPort ,
            warnings: [],
            errors: []
            };
}
break;
    //command
case 12:
    {
        rst[arrayIndex++] = obj.msgId;
        if(obj.cmd >= 0 && obj.cmd <= 9){
            rst[arrayIndex] = obj.cmd;
                    return {
            bytes: rst,
            fPort: fPort ,
            warnings: [],
            errors: []
            };
        }
    }
```

```
    break;
    //ack
case 13:
    {
        rst[arrayIndex] = obj.msgId;
                    return {
          bytes: rst,
          fPort: fPort ,
          warnings: [],
          errors: []
           };
    }
    break;
  //Positioning beacon UUID setting
case 14:
    //Asset beacon UUID setting
case 15:
    {
          var uuidNum = obj.uuidList.length;
          rst[arrayIndex++] = obj.msgId;
          rst[arrayIndex++] = uuidNum;
          if(uuidNum >= 1 && uuidNum <=5){
             for(var i=0; i<uuidNum; i++){
                var uuidIndex = obj.uuidList[i].index;
                if(uuidIndex >= 0 && uuidIndex <= 4){
                   var uuid = obj.uuidList[i].uuid;
                   if(uuid.length == 32){
                      rst[arrayIndex++] = uuidIndex;
                      var pos=0;
                      for(var j=0; j<16; j++)
                      {
                         var s = uuid.substr(pos, 2);
                         var v = parseInt(s, 16);
                         rst[arrayIndex++] = v;
                         pos += 2;
                      }
                   }
                   else{
                                  return {
          bytes: [],
          fPort: fPort ,
          warnings: [],
          errors: []
           };
```

```
                    }
                  }
                }
                        return {
            bytes: rst,
            fPort: fPort ,
            warnings: [],
            errors: []
             };
            }
        }
        break;
    case 16:
        {
            rst[arrayIndex++] = obj.msgId;
            var filterPort = obj.port;
            if(filterPort >= 21 && filterPort <= 25){
                var filterLen = obj.filterLen;
                var filter = obj.filter;
                if(filter.length == (filterLen<<1)){
                  rst[arrayIndex++] = filterPort;
                  rst[arrayIndex++] = obj.start;
                  rst[arrayIndex++] = obj.end;
                  rst[arrayIndex++] = obj.filterStart;
                  rst[arrayIndex++] = obj.filterLen;
                  var filterPos=0;
                  for(var j1=0; j1<filterLen ; j1++)
                  {
                    var s1 = filter.substr(filterPos, 2);
                    var v1 = parseInt(s1, 16);
                    rst[arrayIndex++] = v1;
                    filterPos += 2;
                  }
                }
                        return {
            bytes: rst,
            fPort: fPort ,
            warnings: [],
            errors: []
             };
              }
          }
        break;
    case 17:
```

```
            {
                var beaconNum = obj.number;
                rst[arrayIndex++] = obj.msgId;
                rst[arrayIndex++] = beaconNum;
                if(beaconNum >= 1 && beaconNum <=20){
                  for(var beaconI=0; beaconI<beaconNum; beaconI++){
                    var beaconIndex = obj.beaconList[beaconI].index;
                    if(beaconIndex >= 0 && beaconIndex <= 19){
                        var beaconMajor= obj.beaconList[beaconI].major;
                        var beaconMinor= obj.beaconList[beaconI].minor;
                        rst[arrayIndex++] = beaconIndex;
                        rst[arrayIndex++] =    beaconMajor >> 8;
                        rst[arrayIndex++] = beaconMajor & 0xff;
                        rst[arrayIndex++] =    beaconMinor >> 8;
                        rst[arrayIndex++] = beaconMinor & 0xff;
                      }
                  }
                return {
                  bytes: rst,
                  fPort: fPort ,
                  warnings: [],
                  errors: []
                  };
                }
            }
            break;
          default:
        }
                return {
                  bytes: [],
                  fPort: fPort ,
                  warnings: [],
                  errors: []
                  };
}
    function decodeDownlink(input) {
      return {
        data: {
          bytes: input.bytes
        },
        warnings: [],
        errors: []
      }
}
```

3 Message in JSON

3.1 Uplink Message

This chapter lists the decoded JSON format of each message.

3.1.1 Heartbeat

```
{
        "bleAck":1,
        "code":0,
        "msgType":"heartbeat",
        "parameters":
        {
                "ble":
                {
                        "auReport":0,
                        "bleOff":0,
                        "period":10,
                        "scale":3,
                        "scan":1,
                        "stepsOff":5
                },
                "dr":2,
                "gnssPeriod":30,
                "heartBeatPeriod":300,
                "scheme":3,
                "sleepy":
                {
                        "degree":0,
                        "end":7,
                        "start":22
                },
                "timestamp":1644556501, //Current time in device
                "txPower":1,
                "warning":{
                        "buzzer":0,
                        "distance":6,
```

```
                        "proximity":0,
                        "vibrator":0
                }
        },
        "signal":
        {
                "rssi":-27,
                "snr":10.0
        },
        "status":
        {
                "battery":0, //Charging status
                "gnss":0,
                "soc":100, //Remaining battery
                "vibstate":0
        },
        "steps":60,
        “temp”:22,
        "thres":0, //RSSI detection threshold for beacons      "version":
        {
                "hardwareType":0,
                "swVer":"1.5"
        },
        "workMode":0
}
```


3.1.2 GNSS Coordinate


```
{
        "latitude":
        {
                "orientation":"N", //N: North latitude, S: South latitude
                "value":32.0818933
        },
        "longitude":
        {
                "orientation":"E", //E: East longitude, W: West longitude
                "value":118.8034133
        },
        "msgType":"GNSS coordinate",
        "time":4 // The time taken for this positioning from startup to success
}
```

3.1.3 BLE Coordinate


```
{
        "beaconList":[
        {
                "battery":100,
                "major":"3001",
                "minor":"3000",
                "rssi":-68,
                "type":0
        },
        {
                "battery":99,
                "major":"3001",
                "minor":"3001",
                "rssi":-75,
                "type":0
        },
        {
                "battery":98,
                "major":"2001",
                "minor":"2000",
                "rssi":-75,
                "type":0
        }
        ],
        "code":0,
        "msgType":"BLE coordinate",
        "closecontact":1, //indicates whether the badge is close to others.
        "step":9710
}
```


3.1.4 Alarm


```
{
        "ack":0,
        "alarm":2, //alarm type
        "msgId":0,
        "msgType":"alarm"
}
```

3.1.5 Ack

```
{
        "code":0,
        "msgId":0,
        "msgType":"ack",
        "result":0
}
```

3.1.6 Positioning UUID List

```
{
        "code":0,
        "msgType":"Positioning UUID list",
        "uuidList":[
        {
                "index":0,
                "uuid":"22222222222222222222222222222222"
        },
        {
                "index":1,
                "uuid":"33333333333333333333333333333333"
        },
        {
                "index":2,
                "uuid":"44444444444444444444444444444444"
        },
        {
                "index":3,
                "uuid":"55555555555555555555555555555555"
        },
        {
                "index":4,
                "uuid":"66666666666666666666666666666666"
        }
        ]
}
```

3.1.7 Asset UUID List

```
{
        "code":0,
```

```
        "msgType":"Asset UUID list",
        "uuidList":[
        {
                "index":0,
                "uuid":"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
        },
        {
                "index":1,
                "uuid":"bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"
        },
        {
                "index":2,
                "uuid":"cccccccccccccccccccccccccccccccc"
        },
        {
                "index":3,
                "uuid":"dddddddddddddddddddddddddddddddd"
        },
        {
                "index":4,
                "uuid":"eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee"
        }
        ]
}
```


3.1.8 Pass-through Filter List


```
{
        "code":0,
        "filterList":[
        {
                "end":30,
                "filter":"02a6fd0110004653",
                "filterLen":8,
                "filterStart":7,
                "port":21,
                "start":0
        }
        ],
        "length":1,
        "msgType":"Pass-through filter list"
}
```

3.1.9 History Beacon Config List

The beacons need to be confirmed by the network server, else will be stored in device and try later.


```
{
        "beaconList":[
        {
                "index":0,
                "major":"2001",
                "minor":"2000"
        },
        {
                "index":1,
                "major":"2001",
                "minor":"2001"
        },
        {
                "index":2,
                "major":"3001",
                "minor":"3000"
        },
        {
                "index":3,
                "major":"3001",
                "minor":"3001"
        }
        ],
        "code":0,
        "msgType":"History Beacon Config List",
        "number":4
}
```


3.1.10 History Beacon Info List


```
{
        "beaconList":[
        {
                "frmOff":3,
                "major":"1101",
                "minor":"110e",
                "rssi":-85,
                "timeOff":50
        }
        ],
```

```
        "code":0,
        "msgType":"History Beacon Info List",
        "number":1
}
```


3.1.11 History GNSS Info List


```
{
        "code":0,
        "gnssList":[
        {
                "frmOff":9,
                "latitude":
                {
                        "orientation":"N",
                        "value":32.0813933
                },
                "longitude":
                {
                        "orientation":"E",
                        "value":118.8016
                },
                "timeoff":141
        },
        {
                "frmOff":8,
                "latitude":
                {
                        "orientation":"N",
                        "value":32.0821467
                },
                "longitude":
                {
                        "orientation":"E",
                        "value":118.80228
                },
                "timeoff":101
        }
        ],
        "msgType":"History GNSS Info List",
        "number":2
}
```

3.1.12 Pass-through Data List

```
{
        "code":0,
        "dataList":[
        {
                "index":0,
                "payload":"0201061bffa60202a6fd011000465392deae3f6b83a6d5450c61aaaabbbbbf"
        }
        ],
        "msgType":"Pass-through data list"
}
```

3.2 Downlink Message

The data is provided in JSON format and encoded into binary data. You can send the following JSON data to control the device.

3.2.1 Parameters Setting

3.2.1.1 Tx Power

```
{
        "txPower":1, //0,1,2,3
        "msgId":2  //The application server accumulates the msgId to 0 at 255.
}
```

3.2.1.2 Data Rate

```
{
        "msgId":4,
        "dr":2 //0,1,2. For US915, 0:DR3,1:DR2,2:DR1. For others,0:DR5,1:DR4,2:DR3
}
```

3.2.1.3 AUREPORT

```
{
        "msgId":5,
        "bleAuReport":"enable" //enable,disable
}
```

3.2.1.4 BLE

```
{
        "blePeriod":20,// 0,5,10,20,30,60,120,300,600,900,1200,1800,3600,7200,21600,43200 seconds
        "msgId":7
}
```

3.2.1.5 SCAN

```
{
        "msgId":8,
        "bleScan":2 //1,2,3,6,9,12,15
}
```

3.2.1.6 SCALE

```
{
        "scale":0, //0,1,2,3
        "msgId":9
}
```

3.2.1.7 BLEOFF

```
{
        "bleBleOff":2,// 0,1,2,3,4,5,6,7
        "msgId":10
}
```

3.2.1.8 STEPSOFF

```
{
        "bleStepsOff":3,// 0,1,2,3,4,5,6,7
        "msgId":11
}
```

3.2.1.9 BUZZER

```
{
        "warnBuzzer":"enable", //enable, disable
        "msgId":13
}
```

3.2.1.10 VIBRATOR

```
{
        "warnVibrator":"enable", //enable, disable
        "msgId":14
}
```

3.2.1.11 DISTANCE

```
{
        "warnDistance":4, //2,4,6,8,10,15 meters
        "msgId":15
}
```

3.2.1.12 PROXIMITY

```
{
        "warnProximity":"enable", //enable,disable
        "msgId":12
}
```

3.2.1.13 GNSS Period

```
{
        "gnssPeriod":60, //0,10,20,30,60,120,300,600,1800,3600,7200,10800,21600,43200 seconds
        "msgId":17
}
```

3.2.1.14 HEARTBEAT

```
{
        "msgId":18,
```

```
        "heartBeatPeriod":600 //60,300,600,1200,1800,3600,7200,21600,43200,86400 seconds
}
```

3.2.1.15 DATETIME

Deprecated.

```
{
        "msgId":1,
        "datetime":18623413 //Time stamp, from January 1, 1970, seconds
}
```

3.2.1.16 SLEEPY

```
{
        "msgId":19,
        "sleepy":
        {
                "degree":1,
                "end":6, //hour, UTC time
                "start":21 //hour, UTC time
        }
}
```

3.2.1.17 THRES

```
{
        "msgId":20,
        "thres":2 //0,1,2,3,4,5,6,7
}
```

3.2.1.18 BLEACK

```
{
        "bleAck":"enable", //enable, disable
        "msgId":21
}
```

3.2.1.19 Multiple parameters

Multiple parameters can be set at the same time. For example:

```
{
        "warnProximity":"disable",
        "warnVibrator":"disable",
        "bleBleOff":1,
        "warnBuzzer":"disable",
        "scale":3,
        "msgId":22,
        "dr":1,
        "txPower":2,
        "blePeriod":10,
        "bleAck":"enable",
        "bleStepsOff":2,
        "warnDistance":2,
        "gnssPeriod":30,
        "bleAuReport":"disable",
        "bleScan":1,
        "sleepy":
        {
                "degree":0,
                "end":7,
                "start":23
        },
        "heartBeatPeriod":300,
        "thres":0
}
```


3.2.2 BLE Start Time


```
{
        “time": 3,
        "msgId": 22
}
```


3.2.3 Command


```
{
        "cmd": 2, //0,1,2,3,4,5,6,7,8,9
        "msgId", 23
}
```

3.2.4 Ack

```
{
        "msgId": 24 //Should be the same with the confirmed uplink message.
}
```

3.2.5 Configure locator beacon UUID

```
{
        "msgId":23,
        "uuidList":       [
        {
                "index":0,  //index should be between 0 and 4.
                "uuid":"22222222222222222222222222222222"
        },
        {
                "index":1,
                "uuid":"33333333333333333333333333333333"
        },
        {
                "index":2,
                "uuid":"44444444444444444444444444444444"
        },
        {
                "index":3,
                "uuid":"55555555555555555555555555555555"
        },
        {
                "index":4,
                "uuid":"66666666666666666666666666666666"
        }
        ]
}
```

3.2.6 Configure asset beacon UUID

```
{
        "msgId":24,
        "uuidList":[
        {
                "index":0,
                "uuid":"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
```

```
        },
        {
                "index":1,
                "uuid":"bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"
        },
        {
                "index":2,
                "uuid":"cccccccccccccccccccccccccccccc"
        },
        {
                "index":3,
                "uuid":"dddddddddddddddddddddddddddddd"
        },
        {
                "index":4,
                "uuid":"eeeeeeeeeeeeeeeeeeeeeeeeeeeeee"
        }]
}
```

3.2.7 Configure Pass-through BLE Filter

```
{
        "filter":"02a6fd0110004653",
        "filterStart":7,
        "port":21,
        "start":15,
        "filterLen":8,
        "end":30
}
```

3.2.8 Configure Confirm needed Beacon

```
{
        "number":4,  //Up to 20 beacons can be configured
        "beaconList":[
        {
                "major":8193,
                "minor":8192,
                "index":0
        },
        {
                "major":8193,
                "minor":8193,
```

```
                "index":1
        },
        {
                "major":12289,
                "minor":12288,
                "index":2
        },
        {
                "major":12289,
                "minor":12289,
                "index":3
        }]
}
```


The End