Decoder LoRaWAN – container tracker (TTN/Chirpstack)

Decoders · https://www.lansitec.com/es/docs/decoders/decoder-lorawan-container-tracker-ttn-chirpstack/

decode-container-tracker-ttn_chirpstackv4.jsDownload

Below the full decoder code:

```
// Decode uplink function.
//
// Input is an object with the following fields:
// - bytes = Byte array containing the uplink payload, e.g. [255, 230, 255, 0]
// - fPort = Uplink fPort.
// - variables = Object containing the configured device variables.
//
// Output must be an object with the following fields:
// - data = Object representing the decoded payload.

// Container Tracker
function decodeUplink(input) {
  // bytes
  var bytes = input.bytes;
  // type
  var uplinkType = (bytes[0] >> 4) & 0x0f;

  switch (uplinkType) {
    case 0x01:
      return { data: decodeRegistration(bytes) };

    case 0x02:
      return { data: decodeHeartbeat(bytes) };

    case 0x03:
      return { data: decodeGNSSPosition(bytes) };

    case 0x06:
      return { data: decodeMovementThreshold(bytes) };

    case 0x07:
      return { data: decodePositionBeacon(bytes) };
```

```
    case 0x08:
      return { data: decodeAssetBeacon(bytes) };

    case 0x09:
      return { data: decodeAlarm(bytes) };

    case 0x0a:
      return { data: decodeVibrationShockDetectionReport(bytes) };

    case 0x0b:
      return { data: decodeOfflineCachePosition(bytes) };

    default:
      return null;
  }
}
// type: 0x1 Registration
function decodeRegistration(bytes) {
  var data = {};
  data.type = "Registration";
  data.adr = ((bytes[0] >> 3) & 0x1) == 0 ? "OFF" : "ON";
  // mode
  data.mode = 0x00;
  // smode
  data.smode = 0x00;
  // power
  data.power = ((bytes[2] >> 3) & 0x1f) + "dBm";
  // offlineCacheEnable
  data.offlineCacheEnable =
    ((bytes[2] >> 2) & 0x01) == 0 ? "Disable" : "Enable";
  // alarmEnable
  data.alarmEnable = ((bytes[2] >> 1) & 0x01) == 0 ? "Disable" : "Enable";
  // singleKeyEnable
  data.singleKeyEnable = (bytes[2] & 0x01) == 0 ? "Disable" : "Enable";
  // dr
  data.dr = (bytes[3] >> 4) & 0x0f;
  // gnssEnable
  data.gnssEnable = ((bytes[3] >> 3) & 0x01) == 0 ? "Disable" : "Enable";
  // positionReportMode
  var positionReportModeValue = (bytes[3] >> 1) & 0x03;
  if (positionReportModeValue == 0) {
    data.positionReportMode = "Period";
  } else if (positionReportModeValue == 1) {
```

```
  data.positionReportMode = "Autonomous";
} else if (positionReportModeValue == 2) {
  data.positionReportMode = "On-Demand";
}
// switchEnable
data.switchEnable = (bytes[3] & 0x01) == 0 ? "Disable" : "Enable";
// heartbeatReportInterval
data.heartbeatReportInterval =
  (((bytes[4] << 8) & 0xff00) | (bytes[5] & 0xff)) * 30 + "s";
// blePositionReportInterval
data.blePositionReportInterval =
  (((bytes[6] << 8) & 0xff00) | (bytes[7] & 0xff)) * 5 + "s";
// div
data.div = bytes[8] & 0xff;
// bleEnable
data.bleEnable = (bytes[9] & 0x01) == 0 ? "Disable" : "Enable";
// positioningUUID
var positioningUUID = "";
for (let i = 0; i < 4; i++) {
  var byte1 = bytes[10 + 4 * i];
  var byte2 = bytes[11 + 4 * i];
  var byte3 = bytes[12 + 4 * i];
  var byte4 = bytes[13 + 4 * i];

  var part1 = ((byte1 << 8) | byte2)
    .toString(16)
    .padStart(4, "0")
    .toUpperCase();
  var part2 = ((byte3 << 8) | byte4)
    .toString(16)
    .padStart(4, "0")
    .toUpperCase();

  positioningUUID += part1 + part2;
}
data.positioningUUID = positioningUUID;
// accelerometerThreshold
data.accelerometerThreshold = (50 + bytes[26] * 5) * 0.001 + "g";
// version
data.version = (((bytes[27] << 8) & 0xff00) | (bytes[28] & 0xff))
  .toString(16)
  .toUpperCase();
// cfmmsg
data.cfmmsg = bytes[29];
```

```
  // hbCount
  data.hbCount = bytes[30];
  // assetBeaconReportPeriod
  data.assetBeaconReportPeriod = bytes[31] + "min";
  // bluetoothReceivingDuration
  data.bluetoothReceivingDuration = bytes[32] + "s";
  // extraAssetBeaconReportInterval
  data.extraAssetBeaconReportInterval = bytes[33] + "s";
  // assetBeaconUUID
  var assetBeaconUUID = "";
  for (let i = 0; i < 4; i++) {
    var byte1 = bytes[34 + 4 * i];
    var byte2 = bytes[35 + 4 * i];
    var byte3 = bytes[36 + 4 * i];
    var byte4 = bytes[37 + 4 * i];

    var part1 = ((byte1 << 8) | byte2)
      .toString(16)
      .toUpperCase()
      .padStart(4, "0");
    var part2 = ((byte3 << 8) | byte4)
      .toString(16)
      .toUpperCase()
      .padStart(4, "0");

    assetBeaconUUID += part1 + part2;
  }
  data.assetBeaconUUID = assetBeaconUUID;
  // vibrationShockDetectionThreshold
  data.vibrationShockDetectionThreshold = (50 + bytes[50] * 5) * 0.001 + "g";
  // vibrationShockDetectionReportPeriod
  data.vibrationShockDetectionReportPeriod = bytes[51] * 30 + "s";
  // gnssPositionReportInterval
  data.gnssPositionReportInterval =
    (((bytes[52] << 8) & 0xff00) | (bytes[53] & 0xff)) * 5 + "s";

  return data;
}

// type: 0x2 Heartbeat
function decodeHeartbeat(bytes) {
  var data = {};
  data.type = "Heartbeat";
  // snrEnable
```

```
  data.snrEnable = (bytes[0] & 0x0f) == 0 ? "No SNR field" : "SNR field Enable";
  // voltage
  data.voltage = bytes[1] / 100 + 1.5 + "V";
  // rssi
  data.rssi = bytes[2] * -1 + "dBm";
  // snr
  data.snr = (((bytes[3] << 8) & 0xff00) | (bytes[4] & 0xff)) / 100 + "dB";

  // gnssState
  var gnssStateValue = (bytes[5] >> 4) & 0x0f;
  if (gnssStateValue == 0) {
    data.gnssState = "Off";
  } else if (gnssStateValue == 1) {
    data.gnssState = "Boot GNSS";
  } else if (gnssStateValue == 2) {
    data.gnssState = "Locating";
  } else if (gnssStateValue == 3) {
    data.gnssState = "Located";
  } else if (gnssStateValue == 9) {
    data.gnssState = "No signal";
  }
  // moveState
  data.moveState = bytes[5] & 0x0f;
  // temperature
  data.temperature = (((bytes[6] << 8) & 0xff00) | (bytes[7] & 0xff)) + "□";
  // movement
  data.movement = (((bytes[8] << 8) & 0xff00) | (bytes[9] & 0xff)) * 5 + "s";

  return data;
}

// type: 0x03 GNSSPosition
function decodeGNSSPosition(bytes) {
  var data = {};
  data.type = "GNSSPosition";
  // longitude
  let longitude =
    (bytes[1] << 24) | (bytes[2] << 16) | (bytes[3] << 8) | bytes[4];
  data.longitude = hex2float(longitude);

  // latitude
  let latitude =
    (bytes[5] << 24) | (bytes[6] << 16) | (bytes[7] << 8) | bytes[8];
  data.latitude = hex2float(latitude);
```

```
  // time
  let time =
    (bytes[9] << 24) | (bytes[10] << 16) | (bytes[11] << 8) | bytes[12];
  data.time = timestampToTime((time + 8 * 60 * 60) * 1000);

  return data;
}

// type: 0x06 MovementThreshold
function decodeMovementThreshold() {
  var data = {};
  data.type = "MovementThreshold";
  data.thresholdFlag = bytes[1];
  data.movementThreshold = bytes[2];
  data.stationaryThreshold = bytes[3];

  return data;
}

// type: 0x07 PositionBeacon
function decodePositionBeacon(bytes) {
  var data = {};
  data.type = "PositionBeacon";
  data.length = bytes[0] & 0x0f;
  for (let i = 0; i < data.length; i++) {
    var major = (((bytes[6 + 5 * i] << 8) & 0xff00) | (bytes[7 + 5 * i] & 0xff))
      .toString(16)
      .toUpperCase()
      .padStart(4, "0");
    var minor = (((bytes[8 + 5 * i] << 8) & 0xff00) | (bytes[9 + 5 * i] & 0xff))
      .toString(16)
      .toUpperCase()
      .padStart(4, "0");
    var rssi = bytes[10 + 5 * i] - 256 + "dBm";

    data["beacon" + (i + 1)] = major + minor;
    data["rssi" + (i + 1)] = rssi;
  }

  return data;
}

// type: 0x08 AssetBeacon
```

```
function decodeAssetBeacon(bytes) {
  var data = {};
  data.type = "AssetBeacon";
  data.qty = bytes[1] & 0xff;
  for (let i = 0; i < data.qty; i++) {
    var major = (((bytes[2 + 5 * i] << 8) & 0xff00) | (bytes[3 + 5 * i] & 0xff))
      .toString(16)
      .toUpperCase()
      .padStart(4, "0");
    var minor = (((bytes[4 + 5 * i] << 8) & 0xff00) | (bytes[5 + 5 * i] & 0xff))
      .toString(16)
      .toUpperCase()
      .padStart(4, "0");
    var rssi = bytes[6 + 5 * i] - 256 + "dBm";

    data["beacon" + (i + 1)] = major + minor;
    data["rssi" + (i + 1)] = rssi;
  }

  return data;
}

// type: 0x09 Alarm
function decodeAlarm(bytes) {
  var data = {};
  data.type = "Alarm";
  var alarmValue = bytes[1] & 0xff;
  if (alarmValue == 1) {
    data.alarm = "Alarm";
  }
  return data;
}

// type: 0x0B OfflineCachePosition
function decodeOfflineCachePosition(bytes) {
  var data = {};
  // type
  data.type = "OfflineCachePosition";
  // length
  data.length = bytes[0] & 0x0f;
  // cacheDataType
  data.cacheDataType = bytes[1] & 0xff;
  var beaconIndex = 1;
  var gnssIndex = 1;
```

```
var index = 2;
for (let i = 0; i < data.length; i++) {
  var flag = (data.cacheDataType >> (7 - i)) & 0x01;
  if (flag == 0) {
    var major = (((bytes[index] << 8) & 0xff00) | (bytes[index + 1] & 0xff))
      .toString(16)
      .toUpperCase()
      .padStart(4, "0");
    var minor = (
      ((bytes[index + 2] << 8) & 0xff00) |
      (bytes[index + 3] & 0xff)
    )
      .toString(16)
      .toUpperCase()
      .padStart(4, "0");
    var rssi = bytes[index + 4] - 256 + "dBm";

    data["beacon" + beaconIndex] = major + minor;
    data["rssi" + beaconIndex] = rssi;
    beaconIndex = beaconIndex + 1;
    index = index + 5;
  } else if (flag == 1) {
    var gnss = {};
    // longitude
    var longitude =
      (bytes[index] << 24) |
      (bytes[index + 1] << 16) |
      (bytes[index + 2] << 8) |
      bytes[index + 3];
    gnss.longitude = hex2float(longitude);
    // latitude
    var latitude =
      (bytes[index + 4] << 24) |
      (bytes[index + 5] << 16) |
      (bytes[index + 6] << 8) |
      bytes[index + 7];
    gnss.latitude = hex2float(latitude);
    // time
    var time =
      (bytes[index + 8] << 24) |
      (bytes[index + 9] << 16) |
      (bytes[index + 10] << 8) |
      bytes[index + 11];
    gnss.time = timestampToTime((time + 8 * 60 * 60) * 1000);
```

```
      data["longitude" + gnssIndex] = gnss.longitude;
      data["latitude" + gnssIndex] = gnss.latitude;
      data["time" + gnssIndex] = gnss.time;

      gnssIndex = gnssIndex + 1;
      index = index + 12;
    }
  }

  return data;
}

// type: 0x0A VibrationShockDetectionReport
function decodeVibrationShockDetectionReport(bytes) {
  var data = {};
  data.type = "VibrationShockDetectionReport";
  data.vibrationShockCount = ((bytes[1] << 8) & 0xff00) | (bytes[2] & 0xff);
  return data;
}

function hex2float(num) {
  var sign = num & 0x80000000 ? -1 : 1;
  var exponent = ((num >> 23) & 0xff) - 127;
  var mantissa = 1 + (num & 0x7fffff) / 0x7fffff;
  return sign * mantissa * Math.pow(2, exponent);
}

function timestampToTime(timestamp) {
  const date = new Date(timestamp);
  const year = date.getFullYear();
  const month = (date.getMonth() + 1).toString().padStart(2, "0");
  const day = date.getDate().toString().padStart(2, "0");
  const hour = date.getHours().toString().padStart(2, "0");
  const minute = date.getMinutes().toString().padStart(2, "0");
  const second = date.getSeconds().toString().padStart(2, "0");
  return `${year}-${month}-${day} ${hour}:${minute}:${second}`;
}
```