# Decoder LTE – container tracker

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

decode-lte-container-tracker.jsDownload

To create automatically uplink and downlink messages, please refer to this page:

Lansitec Decoder - Live Calculator - Lansitec

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.

// NB-IoT Container Tracker
function decodeUplink(input) {
  // bytes
  var bytes = input.bytes;
  // ascii -> hexString -> byteArray
  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 0x04:
      return { data: decodeBeacon(bytes) };

    case 0x05:
```

```
      return { data: decodeAlarm(bytes) };

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

    default:
      return;
  }
}

// type: 0x01 Registration
function decodeRegistration(bytes) {
  var data = {};
  data.type = "RegistrationMessage";
  data.bleEnable = (bytes[1] >> 7) & 0x01;
  data.gnssEnable = (bytes[1] >> 6) & 0x01;
  data.networkStatusCheckEnable = (bytes[1] >> 5) & 0x01;
  data.powerSwitchEnable = (bytes[1] >> 4) & 0x01;
  data.assetBeaconSortEnable = (bytes[1] >> 1) & 0x01;
  data.gnssFailureReportEnable = bytes[1] & 0x01;
  data.assetManagementEnable = (bytes[2] >> 7) & 0x01;
  data.positionReportMode = (bytes[3] << 6) & 0x0f;
  data.sosAlarmEnable = (bytes[3] >> 5) & 0x01;
  data.fallDetectionAlarmEnable = (bytes[3] >> 4) & 0x01;
  data.specialBeaconnable = (bytes[3] >> 3) & 0x01;
  data.searchAlarmEnable = (bytes[3] >> 1) & 0x01;

  data.fallDetectionThreshold = bytes[5] & 0xff;
  data.heartbeatPeriod = ((bytes[6] << 8) & 0xff00) | (bytes[7] & 0xff);
  data.blePositionReportInterval =
    ((bytes[8] << 8) & 0xff00) | (bytes[9] & 0xff);
  data.blePositionBeaconReceivingDuration = bytes[10] & 0xff;
  data.gnssPositionReportInterval =
    ((bytes[11] << 8) & 0xff00) | (bytes[12] & 0xff);
  data.gnssReceivingDuration = bytes[13] & 0xff;
  data.assetBeaconReportInterval =
    ((bytes[14] << 8) & 0xff00) | (bytes[15] & 0xff);
  data.assetBeaconReceivingDuration = bytes[16] & 0xff;
  data.softwareVersion = ((bytes[17] << 8) & 0xff00) | (bytes[18] & 0xff);
  var imsi = "";
  for (let i = 0; i < 8; i++) {
    imsi = imsi + bytes[19 + i].toString(16).toUpperCase().padStart(2, "0");
  }
  data.imsi = imsi.substring(0, 15);
```

```
  data.messageId = ((bytes[27] << 8) & 0xff00) | (bytes[28] & 0xff);
  return data;
}

// type: 0x02 Heartbeat
function decodeHeartbeat(bytes) {
  var data = {};
  data.type = "HeartbeatMessage";
  var stateBitField = {};
  stateBitField.bleEnable = (bytes[1] >> 7) & 0x01;
  stateBitField.gnssEnable = (bytes[1] >> 6) & 0x01;
  stateBitField.networkStatusCheck = (bytes[1] >> 5) & 0x01;
  stateBitField.powerSwitchEnable = (bytes[1] >> 4) & 0x01;
  stateBitField.assetBeaconSortEnable = (bytes[1] >> 1) & 0x01;
  stateBitField.gnssFailureReportEnable = bytes[1] & 0x01;
  stateBitField.assetManagementEnable = (bytes[2] >> 7) & 0x01;
  stateBitField.posMode = (bytes[3] << 6) & 0x0f;
  stateBitField.sosAlarmEnable = (bytes[3] >> 5) & 0x01;
  stateBitField.fallDetectionAlarmEnable = (bytes[3] >> 4) & 0x01;
  stateBitField.specialBeaconnable = (bytes[3] >> 3) & 0x01;
  stateBitField.searchAlarmEnable = (bytes[3] >> 1) & 0x01;
  data.stateBitField = stateBitField;

  data.batteryVoltage = ((bytes[5] & 0xff) * 0.01 + 2.00) + "V";
  data.batteryLevel = (bytes[6] & 0xff) + "%";
  data.bleReceivingCount = bytes[7] & 0xff;
  data.gnssOnCount = bytes[8] & 0xff;
  data.temperature =
    (((bytes[9] << 8) & 0xff00) | (bytes[10] & 0xff)) / 100 + "°C";
  data.movement = ((bytes[11] << 8) & 0xff00) | (bytes[12] & 0xff);
  data.chargeDuration = ((bytes[15] << 8) & 0xff00) | (bytes[16] & 0xff);
  data.messageId = ((bytes[17] << 8) & 0xff00) | (bytes[18] & 0xff);
  return data;
}

// type: 0x03 GNSS Position
function decodeGNSSPosition(bytes) {
  var data = {};
  data.type = "GNSSPositionMessage";
  data.gnssStatus = bytes[0] & 0x01;
  // longitude
  var longitude = (bytes[1] << 24) & 0xff000000;
  longitude |= (bytes[2] << 16) & 0xff0000;
  longitude |= (bytes[3] << 8) & 0xff00;
```

```
  longitude |= bytes[4] & 0xff;
  data.longitude = hex2float(longitude);
  // latitude
  var latitude = (bytes[5] << 24) & 0xff000000;
  latitude |= (bytes[6] << 16) & 0xff0000;
  latitude |= (bytes[7] << 8) & 0xff00;
  latitude |= bytes[8] & 0xff;
  data.latitude = hex2float(latitude);
  // time
  var time = (bytes[9] << 24) & 0xff000000;
  time |= (bytes[10] << 16) & 0xff0000;
  time |= (bytes[11] << 8) & 0xff00;
  time |= bytes[12] & 0xff;
  data.time = timestampToTime((time + 8 * 60 * 60) * 1000);
  return data;
}

// type: 0x04 Beacon
function decodeBeacon(bytes) {
  var data = {};
  data.type = "BeaconMessage";
  data.beaconType =
    (bytes[0] & 0x01) == 0 ? "PositioningBeacon" : "AssetBeacon";
  data.beaconCount = bytes[1] & 0x0f;
  for (let i = 0; i < data.beaconCount; i++) {
    var index = 2 + 5 * i;
    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" + (i + 1)] = major + minor;
    data["rssi" + (i + 1)] = rssi;
  }

  return data;
}

// type: 0x05 Alarm
```

```
function decodeAlarm(bytes) {
  var data = {};
  data.type = "AlarmMessage";
  var alarmValue = bytes[1] & 0x0f;
  if (alarmValue == 5) {
    data.alarm = "TamperDetection";
  }
  return data;
}

// type: 0x06 Configuration Parameter Response
function decodeConfigParameterResponse(bytes) {
  var data = {};
  data.type = "ConfigurationParameterResponse";
  var parameter = [];
  let byteLength = bytes.length;
  var index = 0;
  while (index + 1 < byteLength) {
    var commandBitField = {};
    var parameterType = bytes[index + 1] & 0xff;
    commandBitField.parameterType = parameterType;
    var commandBitFieldLength = getCommandBitFieldLength(parameterType);
    var parameterValue = getParameterValue(bytes, index, commandBitFieldLength);
    commandBitField.parameterValue = parameterValue;
    commandBitField.name = getParameterName(parameterType);
    commandBitField.parameterDefinition = getParameterDefinition(
      parameterType,
      parameterValue
    );
    index = index + commandBitFieldLength;
    parameter.push(commandBitField);
  }
  data.parameter = parameter;
  return data;
}

// getParameterValue hexString
function getParameterValue(bytes, index, length) {
  var hexString = "";
  for (var i = 2; i <= length; i++) {
    var hex = (bytes[index + i] & 0xff).toString(16).toUpperCase();
    hexString += hex.padStart(2, "0");
  }
  return hexString;
```

```
}

// getCommandBitFieldLength
function getCommandBitFieldLength(parameterType) {
  let lengths = {
    0x00: 3,
    0x01: 3,
    0x02: 3,
    0x03: 3,
    0x04: 3,
    0x05: 2,
    0x06: 2,
    0x07: 2,
    0x0a: 17,
    0x0b: 17,
    0x0e: 9,
    0x1c: 2,
    0x20: 2,
    0x29: 2,
    0x2a: 2,
    0x2b: 2,
    0x2e: 2,
    0x2f: 2,
    0x30: 2,
    0x31: 2,
  };
  return lengths[parameterType] ?? 0;
}

// Parameter Name
function getParameterName(parameterType) {
  let name = {
    0x00: "SoftwareVersion",
    0x01: "HBPeriod",
    0x02: "BlePositionReportInterval",
    0x03: "GNSSPositionReportInterval",
    0x04: "AssetBeaconReportInterval",
    0x05: "BlePositionReceivingDuration",
    0x06: "GNSSPositionReceivingDuration",
    0x07: "AssetBleReceivingDuration",
    0x0a: "PosBeaconUUID",
    0x0b: "AssetBeaconUUID",
    0x0e: "ISMI",
    0x1c: "TamperDetectionEnable",
```

```
    0x20: "PositionReportMode",
    0x29: "AssetManagementEnable",
    0x2a: "GNSSFailureReportEnable",
    0x2b: "AssetBeaconSortEnable",
    0x2e: "PowerSwitchEnable",
    0x2f: "NetworkStatusCheck",
    0x30: "GNSSEnableState",
    0x31: "BleEnable",
  };
  return name[parameterType] ?? "No matching parameter names";
}

// Parameter Definition
function getParameterDefinition(parameterType, parameterValue) {
  var val = parseInt(parameterValue, 16);
  let name = {
    0x00: "SoftwareVersion",
    0x01: val * 30 + "s, The interval of the heartbeat message, unit: 30s",
    0x02: val * 5 + "s, The interval of Bluetooth position repor, unit: 5s",
    0x03: val * 5 + "s, The interval of GNSS position report, unit: 5s",
    0x04: val * 5 + "s, The interval of asset beacons report, unit: 5s",
    0x05:
      val * 1 + "s, The duration of BLE position beacon receiving, unit: 1s",
    0x06: val * 5 + "s, The duration of GNSS position receiving, unit: 5s",
    0x07: val * 1 + "s, The duration of asset beacon receiving, unit 1s",
    0x0a: "PosBeaconUUIDFilter",
    0x0b: "AssetBeaconUUIDFilter",
    0x0e: "ISMI",
    0x1c: val == 0 ? "Disable" : "Enable",
    0x20:
      val == 0
        ? "Period Mode"
        : val == 1
        ? " Autonomous Mode"
        : "On-demand Mode",
    0x29: val == 0 ? "Disable" : "Enable",
    0x2a: val == 0 ? "Disable" : "Enable",
    0x2b: val == 0 ? "Disable" : "Enable",
    0x2e: val == 0 ? "Disable" : "Enable",
    0x2f: val == 0 ? "Disable" : "Enable",
    0x30: val == 0 ? "Disable" : "Enable",
    0x31: val == 0 ? "Disable" : "Enable",
  };
  return name[parameterType] ?? "No matching parameter names";
```

```
}

function getDeviceState(val) {
  var byteArray = hexStringToByteArray(val);
  var data = {};
  data.bleEnable = (byteArray[0] >> 7) & 0x01;
  data.gnssEnable = (byteArray[0] >> 6) & 0x01;
  data.networkStatusCheck = (byteArray[0] >> 5) & 0x01;
  data.powerSwitchEnable = (byteArray[0] >> 4) & 0x01;
  data.assetBeaconSortEnable = (byteArray[0] >> 1) & 0x01;
  data.gnssFailureReportEnable = byteArray[0] & 0x01;
  data.assetManagementEnable = (byteArray[1] >> 7) & 0x01;
  data.posMode = (byteArray[2] << 6) & 0x0f;
  data.sosAlarmEnable = (byteArray[2] >> 5) & 0x01;
  data.fallDetectionAlarmEnable = (byteArray[2] >> 4) & 0x01;
  data.specialBeaconnable = (byteArray[2] >> 3) & 0x01;
  data.searchAlarmEnable = (byteArray[2] >> 1) & 0x01;
  return data;
}

// Floating point conversion
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 asciiToHex(str) {
  var hexString = "";
  for (let i = 0; i < str.length; i++) {
    var hex = str.charCodeAt(i).toString(16);
    hexString += hex.padStart(2, "0");
  }
  return hexString;
}

function hexStringToByteArray(hexString) {
  var byteArray = [];
  for (let i = 0; i < hexString.length; i += 2) {
    byteArray.push(parseInt(hexString.substr(i, 2), 16));
  }
  return new Uint8Array(byteArray);
}
```

```
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}`;
}
```