Asset Tracker 3 Codec

NOTICE

This document contains proprietary and confidential material of Abeeway SAS. This document is provided under and governed by either a license or a confidentiality agreement. Any unauthorized reproduction, use, or disclosure of this material, or any part thereof, is strictly prohibited.

The material provided in this document is believed to be accurate and reliable. However, no responsibility is assumed by Abeeway SAS for the use of this material. Abeeway SAS reserves the right to make changes to the material at any time and without notice. This document is intended for information and operational purposes only. No part of this document shall constitute any contractual commitment by Abeeway SAS.

REVISIONS

Version	Primary Author(s)	Description of Version	Date Completed
1.0.0	Hend Affes	Initial version	June 20, 2024
2.0.0	Hend Affes	Updated for AT3 firmware v1.15: new notification types (Heartbeat, Shutdown, Data Buffering, FUOTA), new position types (MT3333 GNSS), Query messages, new response types, extended downlink commands and requests, Answer messages, Telemetry notifications, cellular network header, TELEMETRY parameter group	March 17, 2026

Version	Primary Author(s)	Description of Version	Date Completed
2.1.0	Hend Affes	Documentation review against AT3 reference guide v1.5-0: corrected batteryLevel type and special values, frameNumber range, latitude/longitude types, EHPE range encoding, gnssFailure structure, Query field names; added missing BLE parameter group, 8 SYSTEM_CORE params, 7 CELLULAR params, recovery beacon response types; fixed Network/LoRaWAN static-motion parameter name splits, renamed lorawanDlTriggerPeriod→lorawanHeartbeatPeriod and cellCnxNwLostTimeout→cellCnxNwReconnectTimeout; corrected all 29 Telemetry parameter names to camelCase; added AUTHENTICATION_ERROR FUOTA failure reason, AnswerType definitions	June 24, 2026

Version	Primary Author(s)	Description of Version	Date Completed
2.2.0	Hend Affes	Updated for AT3 firmware v1.6: shock notification GADD index widened from 1 to 4 bytes (internal format change, decoded output unchanged); position type mask extended to 5 bits; bleScanNbBeacons max increased to 100; bleBeaconFastAdvInterval and bleBeaconSlowAdvInterval minimum raised to 40 ms; added new CELLULAR parameter cellLpmRecoveryTimeout; added new BLE parameters bleScanPresencePeriod and bleScanPresenceMaxTime	September 7, 2026

Table of Contents

1. Introduction
2. Accessing the Abeeway Codec using ThingPark X Location Engine (SaaS)
    - 2.1 Create a free account
    - 2.2 Generate a token
    - 2.3 Using the Abeeway Codec hosted in ThingPark X Location Engine (SaaS)
    - 2.4 Retrieving the version of the codec installed on the SaaS platform
3. Standalone Installation
    - 3.1 Prerequisites
    - 3.2 Install and build
    - 3.3 ChirpStack integration
    - 3.4 Usage of the Abeeway Codec
        - 3.4.1 Node.js module usage
        - 3.4.2 ChirpStack module usage
        - 3.4.3 Browser module usage
4. Decoding an uplink
    - 4.1 Description
        - 4.1.1 Header object
        - 4.1.2 Extended Header object
        - 4.1.3 Notification object
            - 4.1.3.1 System object
            - 4.1.3.2 Accelerometer object
            - 4.1.3.3 Network object
            - 4.1.3.4 Geozoning object

        * 4.1.3.5 Telemetry object
    - 4.1.4 Position object
        * 4.1.4.1 GNSS Fix object
    - 4.1.5 Query object
    - 4.1.6 Response object
        * 4.1.6.1 Parameter class configuration set object
        * 4.1.6.2 Parameter class configuration get object
        * 4.1.6.3 Generic configurations get object
        * 4.1.6.4 Generic configuration set object
        * 4.1.6.5 CRC configuration response object
        * 4.1.6.6 Sensor request response object
        * 4.1.6.7 Debug information response object
        * 4.1.6.8 FUOTA response object
        * 4.1.6.9 Recovery beacon key update response object
        * 4.1.6.10 Recovery beacon root key index get response object
    - 4.1.7 Parameter configuration
        * 4.1.7.1 Group
        * 4.1.7.2 Internal group
        * 4.1.7.3 System core group
        * 4.1.7.4 Geoloc group
        * 4.1.7.5 GNSS group
        * 4.1.7.6 LR11xx group
        * 4.1.7.7 BLE scan group
        * 4.1.7.8 Accelerometer group
        * 4.1.7.9 Network group
        * 4.1.7.10 LoRaWAN group
        * 4.1.7.11 Cellular group
        * 4.1.7.12 BLE group
        * 4.1.7.13 Telemetry group
    - 4.1.8 Telemetry uplink object
        * 4.1.8.1 Metadata payload
        * 4.1.8.2 Timeseries payload
        * 4.1.8.3 measurementConfig object
        * 4.1.8.4 Measurements array
        * 4.1.8.5 Context array
5. Decoding multiple payloads
6. Encoding a downlink
    - 6.1 Commands
    - 6.2 Requests
        - 6.2.1 Generic configuration set request
        - 6.2.2 Parameter class configuration set request
        - 6.2.3 Generic class configuration get request
        - 6.2.4 Parameter class configuration get request
        - 6.2.5 BLE connectivity status request
        - 6.2.6 CRC configuration request
        - 6.2.7 Get sensor values request
        - 6.2.8 Get debug information request
        - 6.2.9 FUOTA request
        - 6.2.10 Recovery beacon key update request
        - 6.2.11 Recovery beacon root key index get request

- 6.3 Answers

7. Decoding a downlink

1. Introduction

This document describes the Abeeway Firmware Codec (Encoder/Decoder) for the Abeeway Asset Tracker 3, firmware version v1.6-0. It provides detailed information on the output format for decoded uplink messages and the input format for encoding downlink messages, enabling seamless communication with the Abeeway device.

The codec is distributed as an open-source npm package, available at https://github.com/abeeway/abeeway-codec. It is also hosted within the ThingPark X Location Engine (SaaS), where it processes messages efficiently. Client applications interact with the codec to decode uplink payloads and encode downlink commands, ensuring reliable and accurate data exchange.

---

2. Accessing the Abeeway Codec using ThingPark X Location Engine (SaaS)

The easiest way to use the Abeeway codec is to use ThingPark X Location Engine hosted in SaaS. The section below describes the steps needed to access the codec hosted in ThingPark X Location Engine.

The driver used by ThingPark X Location Engine to decode and encode AT3 payloads is publicly available at: https://github.com/actility/device-catalog/tree/main/vendors/abeeway/drivers/asset-tracker-3

This driver can also be used standalone, independently of ThingPark X Location Engine — for example as a Node.js module. See Section 3 — Standalone Installation for details.

2.1 Create a free account

In this step, the user needs to create a free account on ThingPark community portal, https://community.thingpark.org/.

2.2 Generate a token

In this step, the user needs to generate a token that can be used with the DX Location API. The DX Admin API page on ThingPark community portal can be found at: `https://dx-api.thingpark.io/admin/latest/swagger`

The user needs to set `grant_type` to `client_credentials` and enter the login/password in `client_id` and `client_secret` respectively.

Once the command is executed, a response code of `200` should be received and the token is set.

2.3 Using the Abeeway Codec hosted in ThingPark X Location Engine (SaaS)

Once the token is set, switch to the codec module of ThingPark Location API to encode/decode Abeeway tracker payloads. The codec module within DX location API can be accessed from: `https://dx-api.thingpark.io/location-engine/v1/swagger-ui/?urls.primaryName=TPX-LE%20Driver%20API`

Use the `POST /decode` endpoint with the following JSON body structure:

```
{
  "direction": "uplink",
  "meta": {
    "lorawan": { "fPort": 18 },
    "driverId": "abeeway:asset-tracker-3:1"
```

```
  },
  "thing": {
    "model": {
      "application": { "producerId": "abeeway", "moduleId": "asset-tracker-3" },
      "producerId": "abeeway",
      "moduleId": "Compact Tracker",
      "version": "2"
    }
  },
  "raw": "0884830184179961ef4b57"
}
```


2.4 Retrieving the version of the codec installed on the SaaS platform

Use the following command to retrieve the Abeeway Asset Tracker codec version installed on the SaaS platform:

```
curl -X 'GET' 'https://community.thingpark.io/iot-flow/v1/drivers' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer YOUR_TOKEN_HERE' \
  | jq '.[] | select(.id=="abeeway:asset-tracker-3:3") | .version'
```

---

3. Standalone Installation

3.1 Prerequisites

- Node.js >= 16
- npm >= 8

3.2 Install and build

```
# Clone the repository
git clone https://github.com/abeeway/abeeway-codec.git
cd abeeway-codec/AT3

# Install dependencies
npm install

# Build all artifacts
npm run build
```

The build outputs are written to the `dist` folder:

File	Format	Intended usage
`abw-at3-drv.js`	UMD, minified	General-purpose build for CommonJS/Node.js and direct browser script usage
`abw-at3-drv-src.js`	UMD, non-minified	Same as above, but easier to inspect and debug

File	Format	Intended usage
`abw-at3-drv.mjs`	ESM, minified	ES module build for `import`-based usage in browsers or modern runtimes
`abw-at3-drv-chirpstack.js`	UMD, minified, with ChirpStack global function shim	ChirpStack codec script

3.3 ChirpStack integration

Upload `dist/abw-at3-drv-chirpstack.js` to ChirpStack:

`Device Profiles → <your profile> → Codec → Custom JavaScript codec functions → paste → Submit`

The bundle exposes `decodeUplink`, `decodeDownlink`, and `encodeDownlink` as global functions, as required by ChirpStack's JavaScript codec interface.

3.4 Usage of the Abeeway Codec

3.4.1 Node.js module usage

```
const driver = require('./dist/abw-at3-drv.js');

const bytes = Array.from(Buffer.from('105e0c248a00000119ff3ff20436458000a10000000a1765', 'hex'));
const result = driver.decodeUplink({
    bytes,
    fPort: 18,
    recvTime: new Date().toISOString()
});
console.log(JSON.stringify(result, null, 2));
```

3.4.2 ChirpStack module usage

```
const vm = require('vm');
const fs = require('fs');
const ctx = vm.createContext({});
vm.runInContext(fs.readFileSync('./dist/abw-at3-drv-chirpstack.js', 'utf8'), ctx);
const bytes = Array.from(Buffer.from('105e0c248a00000119ff3ff20436458000a10000000a1765', 'hex'));
console.log(JSON.stringify(ctx.decodeUplink({ bytes, fPort: 18 }), null, 2));
```

3.4.3 Browser module usage Start the development server:

```
cd AT3
npm run serve
```

Then open `http://localhost:8080/examples/driver-as-browser-module.html` in a browser.

---

4. Decoding an uplink

4.1 Description

The codec processes a LoRaWAN or LTE uplink payload transmitted by an Abeeway Tracker running the AT3 firmware. The input to the codec includes the raw uplink payload, and the LoRa port number

(18 by default), which must be specified in the meta object provided to the codec.

The output of the codec is a JSON-formatted object. Each uplink consists of a common header followed by a data section. The header may include either a single header object or both a header object and an extended header object (in the case of a multi-frame uplink). The content of the data section varies depending on the type of uplink.

On LTE (cellular) networks, a 10-byte cellular network header is prepended before the application header. The codec strips this header automatically before decoding.

4.1.1 Header object This object is integrated into the output decoder for all uplink messages.

Parameter	Type	Description	Supported HW version
multiFrame	Boolean	Set if the uplink is multi-frame	Compact, Combo-compact
type	String	It indicates the uplink type: `NOTIFICATION`, `POSITION`, `QUERY`, `RESPONSE`, `TELEMETRY`	Compact, Combo-compact
sos	Boolean	Set if the device SOS has been entered	Compact, Combo-compact
ackToken	Number	Acknowledgement token used to inform the server of the reception of LoRa downlink messages	Compact, Combo-compact
batteryLevel	Number | String	The remaining power in the battery expressed as a percentage. Returns `"CHARGING"` when the battery is charging, `"UNKNOWN"` when the battery status is unknown, or a number (1–126) representing the remaining battery percentage	Compact, Combo-compact
timestamp	String	The UTC time at which the payload has been built	Compact, Combo-compact
buffering	Boolean	Present and set to `true` when the message was sent using the special (data-buffering) header format with a 4-byte absolute timestamp. Absent in normal single-frame messages	Compact, Combo-compact

4.1.2 Extended Header object The extended header is used only in the multi-frame case.

Parameter	Type	Description	Supported HW version
groupId	Number	Group identifier [0…7]. Incremented each time a new group of uplinks is sent. Used to uniquely identify a group and avoid mixing uplinks from two different groups	Compact, Combo-compact
last	Boolean	Set to indicate the last uplink in the group	Compact, Combo-compact
frameNumber	Number	Frame identifier inside a group [0…15]. Starts at 0 (first uplink) and is incremented for each subsequent uplink in the same group	Compact, Combo-compact

4.1.3 Notification object The decoder output includes the notification object if the uplink type is `NOTIFICATION`.

Parameter	Type	Description	Supported HW version
notificationClass	String	It defines the notification class: `SYSTEM`, `SOS`, `TEMPERATURE`, `GEOZONING`, `ACCELEROMETER`, `NETWORK`, `TELEMETRY`	Compact, Combo-compact
notificationType	String	It defines the notification type for each class. See table below	Compact, Combo-compact
system	Object	System notification object. See System object	Compact, Combo-compact
temperature	Number	Temperature value expressed in degree Celsius	Compact, Combo-compact
accelerometer	Object	Accelerometer notification object. See Accelerometer object	Compact, Combo-compact
network	Object	Network notification object. See Network object	Compact, Combo-compact
geozoning	Object	Geozoning notification object. See Geozoning object	Combo-compact

Parameter	Type	Description	Supported HW version
telemetryMeasurements	Object	Telemetry measurements object. See Telemetry object	Combo-compact

notificationType values per class:

SYSTEM class notification types: - `STATUS` - `LOW_BATTERY` - `BLE_STATUS` - `TAMPER_DETECTION` - `HEARTBEAT` - `SHUTDOWN` - `DATA_BUFFERING_STATUS` - `FUOTA`

SOS class notification types: - `SOS_ON` - `SOS_OFF`

TEMPERATURE class notification types: - `TEMP_HIGH` - `TEMP_LOW` - `TEMP_NORMAL`

ACCELEROMETER class notification types: - `MOTION_START` - `MOTION_END` - `SHOCK`

NETWORK class notification types: - `MAIN_UP` - `BACKUP_UP`

GEOZONING class notification types : - `ENTRY` - `EXIT` - `IN_HAZARD` - `OUT_HAZARD` - `MEETING_POINT`

TELEMETRY class notification types : - `TELEMETRY` - `TELEMETRY_MODE_BATCH`

4.1.3.1 System object This object is included in the notification payload when the notification class is `SYSTEM`.

Parameter	Type	Description	Supported HW version
status	Object	It defines the system status object. See System Status object	Compact, Combo-compact
lowBattery	Object	lowBattery notification object. Attributes: `batteryVoltage` (mV), `consumption` (mAh)	Compact, Combo-compact
bleStatus	Object	BLE status notification object. Attributes: `state` — `BLE_DISCONNECTED` or `BLE_CONNECTED`	Compact, Combo-compact
tamperDetection	Object	Tamper detection object. Attributes: `state` — `CASING_CLOSED` or `CASING_OPEN`	Compact, Combo-compact
heartbeat	Object	Heartbeat object. See Heartbeat object	Compact, Combo-compact
shutdown	Object	Shutdown object. See Shutdown object	Compact, Combo-compact
dataBufferingStatus	Object	Data buffering status object. See Data Buffering Status object	Combo-compact

Parameter	Type	Description	Supported HW version
fuota	Object	FUOTA failure object. See FUOTA failure object	Combo-compact

System Status object

Parameter	Type	Description	Supported HW version
currentTemperature	Number	The current temperature expressed in degree Celsius	Compact, Combo-compact
resetCause	String	The reset cause. See reset cause values below	Compact, Combo-compact
pageId	Number	The page number of the status message (0, 1, 2, or 3)	Compact, Combo-compact

Reset cause values: `AOS_ERROR_NONE`, `AOS_ERROR_HW_NMI`, `AOS_ERROR_HW_FAULT`, `AOS_ERROR_HW_MPU`, `AOS_ERROR_HW_BUS`, `AOS_ERROR_HW_USAGE`, `AOS_ERROR_HW_IRQ`, `AOS_ERROR_HW_WDOG`, `AOS_ERROR_HW_SWDOG`, `AOS_ERROR_HW_BOR`, `AOS_ERROR_SW_ST_HAL_ERROR`, `AOS_ERROR_SW_FREERTOS_ASSERT`, `AOS_ERROR_SW_FREERTOS_TAS` `AOS_ERROR_SW_BLE_ASSERT`, `AOS_ERROR_SW_RTC_FAIL`, `AOS_ERROR_SW_LORA_FAIL`, `AOS_ERROR_SW_DEBUG`, `AOS_ERROR_SW_APP_START`

System STATUS (PAGE 0)

Parameter	Type	Description	Supported HW version
AT3Version	String	AT3 firmware version	Compact, Combo-compact
configurationVersion	String	Configuration version	Compact, Combo-compact
lrHwVersion	Object	LR11xx hardware version object. Attributes: `hardwareVersion`, `hardwareType`, `firmwareVersion`	Compact, Combo-compact
hwBatchId	Number	Hardware batch identifier	Compact, Combo-compact
hwBomId	Number	Hardware BOM identifier	Compact, Combo-compact
maxTemperature	Number	Maximum temperature since the beginning of the product's life, in degree Celsius	Compact, Combo-compact

Parameter	Type	Description	Supported HW version
minTemperature	Number	Minimum temperature since the beginning of the product's life, in degree Celsius	Compact, Combo-compact
motionPercent	Number	Motion percent	Compact, Combo-compact
batteryVoltage	Number	Battery voltage in mV	Compact, Combo-compact
totalConsumption	Number	Total device consumption in mAh	Compact, Combo-compact
cellularConsumption	Number	Cellular consumption in mAh	Compact, Combo-compact
gnssConsumption	Number	GNSS consumption in mAh	Compact, Combo-compact
wifiConsumption	Number	Wi-Fi consumption in mAh	Compact, Combo-compact
lrGnssConsumption	Number	LR11xx GNSS consumption in mAh	Compact, Combo-compact
bleConsumption	Number	BLE consumption in mAh	Compact, Combo-compact
mcuConsumption	Number	MCU consumption in mAh	Compact, Combo-compact
globalCrc	String	CRC of the configuration (all groups except internal)	Compact, Combo-compact

Almanac STATUS (PAGE 1)

Parameter	Type	Description	Supported HW version
lr11xxGpsDate	Number	Highest date (GPS week number) over all GPS almanac entries	Compact, Combo-compact
lr11xxGpsOutdated	Array	Array containing outdated satellite numbers in the LR11xx GPS almanac. Example: `["Satellite 1", "Satellite 11"]`	Compact, Combo-compact
lr11xxGpsGood	Number	Number of GPS satellites matching the highest date	Compact, Combo-compact
lr11xxBeidouDate	Number	Highest date (BeiDou week number) over all BeiDou almanac entries	Compact, Combo-compact

Parameter	Type	Description	Supported HW version
lr11xxBeidouOutdated	Array	Array containing outdated satellite numbers in the LR11xx BeiDou almanac	Compact, Combo-compact
lr11xxBeidouGood	Number	Number of BeiDou satellites matching the highest date	Compact, Combo-compact
gnssGpsDate	Number	Highest GPS date over all MT3333 GPS almanac entries	Compact, Combo-compact
gnssGpsOutdated	Array	Array containing outdated entries in the MT3333 GPS almanac	Compact, Combo-compact
gnssGpsGood	Number	Number of GPS satellites matching the highest date (MT3333)	Compact, Combo-compact
gnssBeidouDate	Number	Highest BeiDou date over all MT3333 BeiDou almanac entries	Compact, Combo-compact
gnssBeidouOutdated	Array	Array containing outdated entries in the MT3333 BeiDou almanac	Compact, Combo-compact
gnssBeidouGood	Number	Number of BeiDou satellites matching the highest date (MT3333)	Compact, Combo-compact

CELLULAR STATUS (PAGE 2)

Parameter	Type	Description	Supported HW version
cellVersion	Object	The cellular version object. Attributes: `branch`, `mode`, `image`, `delivery`, `release` (all numbers)	Compact, Combo-compact
ICCID	String	ICCID identifier	Compact, Combo-compact
IMSI	String	IMSI	Compact, Combo-compact

CELLULAR STATUS (PAGE 3)

Parameter	Type	Description	Supported HW version
EUICCID	String	EUICCID	Compact, Combo-compact
IMEISV	String	IMEISV	Compact, Combo-compact

Heartbeat object This object is included in the system notification payload when the notification type is `HEARTBEAT`.

Parameter	Type	Description	Supported HW version
currentTemperature	Number	Current temperature in degree Celsius	Compact, Combo-compact
resetCause	String	The reset cause. See reset cause values in System Status object	Compact, Combo-compact
globalCrc	String	Global CRC of the configuration (4 bytes hex). Present when config CRC is included	Compact, Combo-compact

Shutdown object This object is included in the system notification payload when the notification type is `SHUTDOWN`.

Parameter	Type	Description	Supported HW version
shutdownCause	String	The shutdown cause: `SHUTDOWN_CAUSE_NONE`, `SHUTDOWN_CAUSE_USER_ACTION`, `SHUTDOWN_CAUSE_LOW_BATTERY`	Compact, Combo-compact

Data Buffering Status object This object is included in the system notification payload when the notification type is `DATA_BUFFERING_STATUS`.

Parameter	Type	Description	Supported HW version
status	String	Data buffering operation status: `SUCCESS`, `TIMEOUT`, `NO_DATA_FOUND`	Combo-compact
oldestTimestamp	String	ISO 8601 UTC timestamp of the oldest buffered message	Combo-compact
latestTimestamp	String	ISO 8601 UTC timestamp of the latest buffered message	Combo-compact

FUOTA failure object This object is included in the system notification payload when the notification type is `FUOTA`.

Parameter	Type	Description	Supported HW version
binary	String	The firmware binary that failed to update: `APPLICATION`, `BLE_STACK`, `MT3333`, `LR11XX`	Combo-compact
source	String	The update source: `XMODEM`, `BLE`, `CELLULAR`, `LORA`	Combo-compact
raison	String	The failure reason: `NONE`, `NETWORK_ISSUE`, `DOWNLOAD_TIMEOUT`, `MODEM_FAILURE`, `FILE_NOT_FOUND`, `PLATFORM_ERROR`, `PARSING_ERROR`, `FLASH_ERROR`, `LENGTH_ERROR`, `SIGNATURE_ERROR`, `AUTHENTICATION_ERROR`	Combo-compact

4.1.3.2 Accelerometer object This object is included in the notification payload when the notification class is `ACCELEROMETER`.

Parameter	Type	Description	Supported HW version
accelerationVector	Array	Acceleration vector with the following format: [X, Y, Z]. X: Acceleration in mg on X axis. Y: Acceleration in mg on Y axis. Z: Acceleration in mg on Z axis	Compact, Combo-compact
motionPercent	Number	Motion percent	Compact, Combo-compact
gaddIndex	Number	GADD index	Compact, Combo-compact
numberShocks	Number	Number of shocks detected	Compact, Combo-compact

Note: Fields present depend on notification type: - `MOTION_START`: no `accelerometer` sub-object (the field is absent from the notification) - `MOTION_END`: `accelerationVector` and `motionPercent` are populated; `gaddIndex` and `numberShocks` are null - `SHOCK`: `accelerationVector`, `gaddIndex`, and `numberShocks` are populated; `motionPercent` is null

4.1.3.3 Network object This object is included in the notification payload when the notification class is `NETWORK`.

Parameter	Type	Description	Supported HW version
activeNetwork	String	The active network: `NO_NETWORK`, `LORA_NETWORK`, `CELLULAR_NETWORK_IN_LOW_POWER_MODE`, `CELLULAR_NETWORK_IN_HIGH_POWER_MODE`	Compact, Combo-compact
mainNetwork	String	The main network. Same values as `activeNetwork`	Compact, Combo-compact
backupNetwork	String	The backup network. Same values as `activeNetwork`	Compact, Combo-compact
psmActiveTime	Number	PSM active time in seconds (present on cellular networks only)	Combo-compact
psmTAU	Number	PSM TAU (Tracking Area Update) timer value in seconds (present on cellular networks only)	Combo-compact

4.1.3.4 Geozoning object This object is included in the notification payload when the notification class is `GEOZONING`. Notification types: `ENTRY`, `EXIT`, `IN_HAZARD`, `OUT_HAZARD`, `MEETING_POINT`.

This notification carries no additional payload fields beyond `notificationClass` and `notificationType`.

4.1.3.5 Telemetry object This object is included in the notification payload when the notification class is `TELEMETRY` and the notification type is `TELEMETRY`. This is a simple single-measurement notification (not the batch timeseries format). For the full timeseries format transmitted as a dedicated TELEMETRY uplink, see Section 4.1.8 — Telemetry uplink object.

The `telemetryMeasurements` field is an object where each key represents an ontology name and the value is an object with `unitId` and `record` attributes.

Example:

```
{
  "telemetryMeasurements": {
    "temperature": { "unitId": "Cel", "record": 23.5 },
    "humidity": { "unitId": "%RH", "record": 65 }
  }
}
```

Supported telemetry ontologies:

Ontology ID	Name	Type	Unit
1	resistance	int16	Ohm
2	temperature	float	Cel (°C)
3	humidity	int16	%RH (divide by 10 for percentage)

If multiple measurements of the same type are reported in one uplink, they are indexed as `temperature:0`, `temperature:1`, etc.

4.1.4 Position object The decoder output includes the position object if the uplink type is `POSITION`.

Parameter	Type	Description	Supported HW version
positionType	String	The position type. See position type values below	Compact, Combo-compact
status	String	The position status: `SUCCESS`, `TIMEOUT`, `FAILURE`, `NOT_SOLVABLE`	Compact, Combo-compact
motion	Number	Set if a motion event has been detected since the transmission of the previous position payload	Compact, Combo-compact
motionCounter	Number	Increments if a motion event has been detected since the transmission of the previous position payload. The counter rolls over once the max value is reached	Compact, Combo-compact
triggers	Object	Triggers are defined in event trigger bit mask	Compact, Combo-compact
gnssFix	Object	The GNSS fix object. See GNSS Fix object	Compact, Combo-compact
gnssFailure	Object	Object with fields `timeoutCause` (String: `T0_TIMEOUT`, `T1_TIMEOUT`, `ACQUISITION_TIMEOUT`) and `satellitesSeen` (Array of objects, each with `svId`, `constellation`, `CN`). Present only when `positionType` is `GNSS` and `status` is `TIMEOUT` or `FAILURE`	Compact, Combo-compact
wifiBssids	Array of objects	An array of (MAC, RSSI) objects. Field populated for successful Wi-Fi scan	Compact, Combo-compact
bleBeaconMacs	Array of objects	An array of (MAC, RSSI) objects. Field populated for successful BLE scan	Compact, Combo-compact

Parameter	Type	Description	Supported HW version
bleBeaconIds	Array of objects	An array of (ID, RSSI) objects. Field populated for successful BLE scan	Compact, Combo-compact
coordinates	Array of double	An array of coordinates as per the GeoJSON specification: [longitude, latitude, altitude]	Compact, Combo-compact

Position type values: - `LR11xx_A_GNSS` — LR1110 GNSS formatted Nav1 (per the firmware reference, this position type carries a decodable satellite table: time, satellite ID, constellation, C/N and pseudo-range. This driver does not decode that structure — the raw bytes are returned as a hex string in `lr11xxAGnss`) - `LR11xx_GNSS_NAV1` — LR1110 GNSS Semtech Nav1 (opaque data produced by the LR1110, meant to be forwarded as-is to the Semtech cloud for decoding — not decodable by this driver. Returned as a hex string in `lr11xxGnssNav1`) - `LR11xx_GNSS_NAV2` — LR1110 GNSS Semtech Nav2 (same as Semtech Nav1: opaque, forwarded as-is. Returned as a hex string in `lr11xxGnssNav2`) - `WIFI` — Wi-Fi BSSIDs - `BLE_SCAN1_MAC` — BLE scan 1, MAC addresses - `BLE_SCAN1_SHORT` — BLE scan 1, short beacon IDs - `BLE_SCAN1_LONG` — BLE scan 1, long beacon IDs - `BLE_SCAN2_MAC` — BLE scan 2, MAC addresses - `BLE_SCAN2_SHORT` — BLE scan 2, short beacon IDs - `BLE_SCAN2_LONG` — BLE scan 2, long beacon IDs - `GNSS` — MT3333 GNSS fix - `AIDED_GNSS` — MT3333 LP-GNSS aided (header recognized; no additional payload fields are currently decoded)

4.1.4.1 GNSS Fix object This object is included in the position object when the position type is `GNSS` and the position status is `SUCCESS`.

Parameter	Type	Description	Supported HW version
latitude	Number	The latitude of the position expressed in degrees	Compact, Combo-compact
longitude	Number	The longitude of the position expressed in degrees	Compact, Combo-compact
altitude	Number	The altitude, signed and expressed in meters	Compact, Combo-compact
COG	Number	Course Over Ground expressed in 1/100 of degrees	Compact, Combo-compact

Parameter	Type	Description	Supported HW version
EHPE	Number | String	Estimated Horizontal Position Error. Values 0–250 are returned as a Number (meters). Values above 250 are returned as a String range: 251 → `"(250,500]"`, 252 → `"(500,1000]"`, 253 → `"(1000,2000]"`, 254 → `"(2000,4000]"`, 255 → `">4000"`	Compact, Combo-compact
SOG	Number	Speed Over Ground expressed in cm/second	Compact, Combo-compact
quality	Object	The fix quality object. Attributes: `fixQuality` (`INVALID`, `VALID`, `FIX_2D`, `FIX_3D`) and `numberSatellitesUsed` (number of satellites used for the fix)	Compact, Combo-compact

4.1.5 Query object The decoder output includes the query object if the uplink type is `QUERY`. Queries are sent by the device to request data from the server (e.g., aiding position, almanac updates, echo test).

Parameter	Type	Description	Supported HW version
queryType	String	The query type: `AIDING_POSITION`, `ECHO_REQUEST`, `UPDATE_GPS_ALMANAC`, `UPDATE_BEIDOU_ALMANAC`. `ECHO_REQUEST` is cellular-only (Combo-compact); the others apply to any GNSS-equipped tracker	Compact, Combo-compact
sequenceNumber	Array of Number	Sequence number bytes (present for `ECHO_REQUEST` only, cellular only)	Combo-compact
gpsSvid	Array of Number	List of GPS satellite IDs requesting almanac update (present for `UPDATE_GPS_ALMANAC`)	Compact, Combo-compact

Parameter	Type	Description	Supported HW version
beidouSvid	Array of Number	List of BeiDou satellite IDs requesting almanac update (present for `UPDATE_BEIDOU_ALMANAC`)	Compact, Combo-compact

4.1.6 Response object The decoder output includes the response object if the uplink type is `RESPONSE`.

Parameter	Type	Description	Supported HW version
responseType	String	The response type. See values below	Compact, Combo-compact
parameterClassConfigurationSet	Object	Refer to parameter class configuration set object	Compact, Combo-compact
parameterClassConfigurationGet	Object	Refer to parameter class configuration get object	Compact, Combo-compact
genericConfigurationGet	Array of objects	Refer to generic configurations get object	Compact, Combo-compact
genericConfigurationSet	Array of objects	Refer to generic configuration set object	Compact, Combo-compact
crc	Object	Refer to CRC configuration response object	Compact, Combo-compact
sensors	Array of objects	Refer to Sensor request response object	Compact, Combo-compact
fuotaStatus	String	Refer to FUOTA response object	Combo-compact
bleStatusConnectivity	String	BLE connectivity status. Present when `responseType` is `BLE_STATUS_CONNECTIVITY`. Values: `IDLE`, `ADVERTISING`, `CONNECTED`, `BONDED`	Compact, Combo-compact
localCrc	String	3-byte local CRC of the configuration group. Present when `responseType` is `PARAM_CLASS_CONFIGURATION_SET`	Compact, Combo-compact
globalCrc	String	4-byte global CRC of the configuration. Present when `responseType` is `GENERIC_CONFIGURATION_SET`	Compact, Combo-compact

Parameter	Type	Description	Supported HW version
recoveryBeaconKeyUpdateStatus	String	Status of the recovery beacon key update. Values: `SUCCESS`, `FAILED_INVALID_ROOT_INDEX`, `FAILED_WRITING_ERROR`, `FAILED_SAVING_ERROR`. Present when `responseType` is `RECOVERY_BEACON_KEY_UPDATE`	Compact, Combo-compact
recoveryBeaconRootKeyIndex	Number	Current root key index (0–7). Present when `responseType` is `RECOVERY_BEACON_ROOT_KEY_INDEX_GET`	Compact, Combo-compact

responseType values: - `GENERIC_CONFIGURATION_SET` - `PARAM_CLASS_CONFIGURATION_SET` - `GENERIC_CONFIGURATION` - `PARAM_CLASS_CONFIGURATION_GET` - `BLE_STATUS_CONNECTIVITY` - `CRC_CONFIGURATION_REQUEST` - `SENSOR_REQUEST` - `DEBUG_INFO_REQUEST` - `FUOTA_REQUEST` - `RECOVERY_BEACON_KEY_UPDATE` - `RECOVERY_BEACON_ROOT_KEY_INDEX_GET`

BLE connectivity status values (responseType = `BLE_STATUS_CONNECTIVITY`): - `IDLE` - `ADVERTISING` - `CONNECTED` (connected but not bonded) - `BONDED`

4.1.6.1 Parameter class configuration set object This object is part of the output decoder for a response to a request that sets configuration parameters within the same class (group). The response type must be `PARAM_CLASS_CONFIGURATION_SET`.

Parameter	Type	Description	Supported HW version
group	String	Refer to group	Compact, Combo-compact
parameters	Array of object	Parameters are an array of objects. Each object has two attributes: `parameterName` (must belong to the group parameters, see parameter configuration) and `status` (local status of the parameter): `SUCCESS`, `NOT_FOUND`, `BELOW_LOWER_BOUND`, `ABOVE_HIGHER_BOUND`, `BAD_VALUE`, `TYPE_MISMATCH`, `OPERATION_ERROR`, `READ_ONLY`	Compact, Combo-compact

Note: The response also carries a top-level `localCrc` field on the Response object (see Response object).

4.1.6.2 Parameter class configuration get object This object is part of the output decoder for the response to a request that retrieves configuration parameters within the same class (group). The response type must be `PARAM_CLASS_CONFIGURATION_GET`.

Parameter	Type	Description	Supported HW version
group	String	Refer to group	Compact, Combo-compact
parameters	Array of object	Parameters are an array of objects. Each object has two attributes: `parameterName` (must correspond to a parameter within the group, see parameter configuration) and `parameterValue` (the reported value for this parameter)	Compact, Combo-compact

4.1.6.3 Generic configurations get object This object is part of the output decoder for the response to a request that retrieves configuration parameters. The response type must be `GENERIC_CONFIGURATION_GET`. `GenericConfigurationGet` is an array of objects where each object represents a group and its parameters.

Parameter	Type	Description	Supported HW version
group	String	Refer to group	Compact, Combo-compact
parameters	Array of object	Each object has two attributes: `parameterName` and `parameterValue` (the reported value for this parameter)	Compact, Combo-compact

4.1.6.4 Generic configuration set object This object is part of the output decoder for the response to a request that sets configuration parameters. The response type must be `GENERIC_CONFIGURATION_SET`. `GenericConfigurationSet` is an array of objects where each object represents a group and its parameters.

Parameter	Type	Description	Supported HW version
group	String	Refer to group	Compact, Combo-compact

Parameter	Type	Description	Supported HW version
parameters	Array of object	Each object has two attributes: `parameterName` and `status` (`SUCCESS`, `NOT_FOUND`, `BELOW_LOWER_BOUND`, `ABOVE_HIGHER_BOUND`, `BAD_VALUE`, `TYPE_MISMATCH`, `OPERATION_ERROR`, `READ_ONLY`)	Compact, Combo-compact

Note: The response also carries a top-level `globalCrc` field on the Response object (see Response object).

4.1.6.5 CRC configuration response object This object is part of the output decoder when the response type is `CRC_CONFIGURATION_REQUEST`.

The `crc` field is an array of strings. Each string is either: - `"Global CRC: 0xXXXXXXXX"` (when a global CRC was requested, bitmap = `0x0000`) - `"GROUP_NAME: 0xXXXXXX"` (when per-group CRCs were requested)

Example:

```
{
  "responseType": "CRC_CONFIGURATION_REQUEST",
  "crc": [
    "SYSTEM_CORE: 0xA1B2C3",
    "GEOLOC: 0xD4E5F6"
  ]
}
```

4.1.6.6 Sensor request response object This object is part of the output decoder when the response type is `SENSOR_REQUEST`.

The `sensors` field is an array of sensor reading objects.

Accelerometer sensor (sensorId = 1):

Parameter	Type	Description
sensorId	Number	Sensor identifier (`1` = accelerometer)
type	String	`ACCELEROMETER`
accelerationVector	Array	[X, Y, Z] acceleration values in mg

Example:

```
{
  "responseType": "SENSOR_REQUEST",
  "sensors": [
    {
```

```
      "sensorId": 1,
      "type": "ACCELEROMETER",
      "accelerationVector": [12, -5, 1020]
    }
  ]
}
```

4.1.6.7 Debug information response object This object is part of the output decoder when the response type is `DEBUG_INFO_REQUEST`. The debug information content depends on the requested info type and is implementation-specific. Refer to the AT3 Reference Guide v1.6-0 for details.

4.1.6.8 FUOTA response object When the response type is `FUOTA_REQUEST`, the `fuotaStatus` field is a string with one of the following values:

Value	Description
`START_ASAP`	FUOTA is supported and will start immediately
`SCHEDULED`	FUOTA is supported and has been scheduled
`DENIED_OPERATION_NOT_SUPPORTED`	No LTE module or external flash missing
`DENIED_CELLULAR_NOT_CONFIGURED`	Missing IP address, URL, or port
`DENIED_SERVER_NOT_CONFIGURED`	No FUOTA server configured
`DENIED_TEMPORARILY_NOT_ALLOWED`	Temporarily denied (e.g., SOS active)
`DENIED_LOW_BATTERY`	Battery too low
`DENIED_LOW_TEMPERATURE`	Temperature too low
`DENIED_HIGH_TEMPERATURE`	Temperature too high
`DENIED_INCORRECT_USER_REQUEST`	Invalid request parameters

4.1.6.9 Recovery beacon key update response object When the response type is `RECOVERY_BEACON_KEY_UPDAT` the `recoveryBeaconKeyUpdateStatus` field contains one of the following values:

Value	Description
`SUCCESS`	Keys updated successfully
`FAILED_INVALID_ROOT_INDEX`	The provided root key index is invalid
`FAILED_WRITING_ERROR`	Error writing keys to storage
`FAILED_SAVING_ERROR`	Error saving keys to persistent storage

4.1.6.10 Recovery beacon root key index get response object When the response type is `RECOVERY_BEACON_ROOT_KEY_INDEX_GET`, the `recoveryBeaconRootKeyIndex` field contains the current root key index as a Number (0–7).

4.1.7 Parameter configuration The configuration parameters are organized into groups based on functionality. Below, we list the groups as defined in the codec, along with the parameters within each group.

Please note that each parameter name must exactly match the spelling as specified in the tables below to be accepted.

4.1.7.1 Group The list of the group (class) parameter.

Parameter	Type	Description	Supported HW version
group	String	The group can be: `INTERNAL`, `SYSTEM_CORE`, `GEOLOC`, `GNSS`, `LR11xx`, `BLE_SCAN1`, `BLE_SCAN2`, `ACCELEROMETER`, `NETWORK`, `LORAWAN`, `CELLULAR`, `BLE`, `TELEMETRY`	Compact, Combo-compact

4.1.7.2 Internal group

group: "INTERNAL"			
Parameter name	Type	Description	Supported HW version
sysHighestTemperature	Number	Highest temperature reached	Compact, Combo-compact
sysLowestTemperature	Number	The lowest temperature reached	Compact, Combo-compact
sysPowerConsumption	Number	Total power consumed	Compact, Combo-compact
sysCellJoinNonce	Number	Cellular join nonce used in ASLP (range: 0–4294967295, default: 0)	Compact, Combo-compact

4.1.7.3 System core group The core group contains the configuration parameters of the system.

group: "SYSTEM_CORE"			
Parameter name	Type	Description	Supported HW version
coreMonitoringPeriod	Number	Device monitoring period	Compact, Combo-compact
coreStatusPeriod	Number	Status monitoring period	Compact, Combo-compact
coreNotifEnable	Object	Notification enables bit mask for classes system, SOS, temperature, accelerometer, network and geozoning. See core notification enable object	Compact, Combo-compact
coreTempHighThreshold	Number	Highest temperature detection threshold	Compact, Combo-compact

group: "SYSTEM_CORE"			
coreTempLowThreshold	Number	Lowest temperature detection threshold	Compact, Combo-compact
coreTempHysteresis	Number	Temperature hysteresis	Compact, Combo-compact
coreButton1Map	Object	Button 1 mapping. See button mapping	Compact, Combo-compact
coreButton2Map	Object	Button 2 mapping. See button mapping	Compact, Combo-compact
coreButtonsTiming	Object	Define the buttons timing parameters. See core buttons timing object	Compact, Combo-compact
coreLed0Map	Object	Defines LED patterns for system events (LED 0)	Compact, Combo-compact
coreLed1Map	Object	Defines LED patterns for system events (LED 1)	Compact, Combo-compact
coreBuzzerMap	Object	Defines buzzer patterns for system events	Compact, Combo-compact
coreAlmanacValidity	Number	Number of days for which the GNSS almanac is considered valid	Compact, Combo-compact
coreAlmanacOutdatedRatio	Number	Percentage of outdated GNSS almanac entries which will trigger a download	Compact, Combo-compact
coreCliPassword	String	User CLI password	Compact, Combo-compact
coreDBTypeMask	Object	Data buffering type mask	Compact, Combo-compact

Core notification enable object

Bit position	Flag Name	Type	Description	Supported HW version
Byte 0	systemClass	Object	systemClass object: Each attribute defines a notification. Attributes: `status` (System status: versions, temperature, reset cause), `lowBattery` (Low battery detected), `bleStatus` (BLE securely connected/disconnected), `tamperDetection` (Tamper detection), `heartbeat` (Heartbeat message), `shutdown` (Shutdown message), `dataBuffering` (Data buffering status), `fuota` (FUOTA failure reason)	Compact, Combo-compact
Byte 1	sosClass	Object	sosClass object: Each attribute defines a notification. Attributes: `sosOn` (SOS activated), `sosOff` (SOS deactivated)	Compact, Combo-compact

Bit position	Flag Name	Type	Description	Supported HW version
Byte 2	temperatureClass	Object	temperatureClass object: Each attribute defines a notification. Attributes: `tempHigh` (Critical high temperature), `tempLow` (Critical low temperature), `tempNormal` (Temperature back to normal)	Compact, Combo-compact
Byte 3	accelerometerClass	Object	accelerometerClass object: Each attribute defines a notification. Attributes: `motionStart` (Motion start detected), `motionEnd` (Motion end detected), `shock` (Shock detected)	Compact, Combo-compact
Byte 4	networkingClass	Object	networkingClass object: Each attribute defines a notification. Attributes: `mainUp` (Main network is up), `backupUp` (Main network down. Backup is up)	Compact, Combo-compact
Byte 5	geozoningClass	Object	geozoningClass object: Each attribute defines a notification. Attributes: `geozoningOn` (Geozoning event detected)	Compact, Combo-compact

Button mapping

Bit position	Flag Name	Type	Description	Supported HW version
Bit 0–3	buttonPress	String	Event generated on a button press: `NO_ACTION`, `DISPLAY_BATTERY_LEVEL_ON_THE_LED`, `START_STOP_SOS`, `REQUEST_A_POSITION_ON_DEMAND`, `FORCE_AN_UPLINK_SYSTEM_STATUS_NOTIFICATION_T` `START_DEVICE`, `STOP_DEVICE`, `START_ONLY_SOS`, `START_BLE_ADVERTISING_FOR_CONNECTIVITY`, `RESET_DEVICE`	Compact, Combo-compact
Bit 4–7	buttonLongPress	String	Event generated on a button long press. Same values as `buttonPress`	Compact, Combo-compact
Bit 8–11	buttonSingleClick	String	Event generated on a button single click. Same values as `buttonPress`	Compact, Combo-compact
Bit 12–15	buttonDoubleClicks	String	Event generated on a button double click. Same values as `buttonPress`	Compact, Combo-compact
Bit 16–19	buttonTripleClicksOrAbove	String	Event generated on a button triple click or above. Same values as `buttonPress`	Compact, Combo-compact
Bit 20–23	buttonSimpleSequence	String	Event generated on a button simple sequence (hard-coded press/release/press pattern). Same values as `buttonPress`	Compact, Combo-compact
Bit 24–31	—	—	Reserved for future use (RFU)	

Core buttons timing object

Bit position	Flag Name	Type	Description	Supported HW version
Bit 0–3	durationButtonPress	Number	Duration of the button press, in units of 0.5 seconds	Compact, Combo-compact
Bit 4–7	durationButtonLongPress	Number	Delay added to the button press duration to detect a long press, in units of 0.5 seconds	Compact, Combo-compact
Bit 8–15	debounceDurationOnButton1	Number	Debounce duration on button 1, in milliseconds	Compact, Combo-compact
Bit 16–23	debounceDurationOnButton2	Number	Debounce duration on button 2, in milliseconds	Compact, Combo-compact
Bit 24–31	—	—	Reserved for future use (RFU)	

4.1.7.4 Geoloc group This group configures the geolocation.

group: "GEOLOC"			
Parameter	Type	Description	Supported HW version
geolocMotionPeriod	Number	Position acquisition period while the device is moving	Compact, Combo-compact
geolocStaticPeriod	Number	Position acquisition period while the device is static	Compact, Combo-compact
geolocSosPeriod	Number	Position acquisition period while the device is in SOS mode	Compact, Combo-compact
geolocMotionNbStart	Number	Number of acquisitions on motion start event	Compact, Combo-compact
geolocMotionNbStop	Number	Number of acquisitions on motion stop event	Compact, Combo-compact
geolocStartStopPeriod	Number	Position acquisition period while acquiring consecutive positions on motion start or stop	Compact, Combo-compact

group: "GEOLOC"			
geolocGnssHoldOnMode	Number	GNSS hold on mode: `DISABLED`, `ALWAYS`, `TECHNO`, `MOVING`, `STATIC`, `TECHNO_AND_MOVING`	Compact, Combo-compact
geolocGnssHoldOnTimeout	Number	GNSS hold on mode timeout	Compact, Combo-compact
geolocProfile0Triggers	Object	Geolocation event triggers 0. See event trigger bit mask	Compact, Combo-compact
geolocProfile1Triggers	Object	Geolocation event triggers 1. See event trigger bit mask	Compact, Combo-compact
geolocProfile2Triggers	Object	Geolocation event triggers 2. See event trigger bit mask	Compact, Combo-compact
geolocGbeProfile0Techno	Array of object	Technologies to schedule using the basic engine for events in triggers 0. Array of geolocation technology objects	Compact, Combo-compact
geolocGbeProfile1Techno	Array of object	Technologies to schedule using the basic engine for events in triggers 1. Array of geolocation technology objects	Compact, Combo-compact
geolocGbeProfile2Techno	Array of object	Technologies to schedule using the basic engine for events in triggers 2. Array of geolocation technology objects	Compact, Combo-compact

Event trigger bit mask

Bit position	Flag Name	Description	Supported HW version
Bit 0	geoTriggerPod	Geoloc triggered on Position-on-demand via downlink or button	
Bit 1	geoTriggerSos	SOS started	
Bit 2	geoTriggerMotionStart	Geoloc triggered on motion start event	
Bit 3	geoTriggerMotionStop	Geoloc triggered on a motion stop event	

Bit position	Flag Name	Description	Supported HW version
Bit 4	geoTriggerInMotion	Periodic geoloc while the tracker is in motion	Compact, Combo-compact
Bit 5	geoTriggerInStatic	Periodic geoloc while the tracker is static	Compact, Combo-compact
Bit 6	geoTriggerShock	Geoloc triggered on a shock action	Compact, Combo-compact
Bit 7	geoTriggerTempHighThreshold	Geoloc triggered on high temperature	Compact, Combo-compact
Bit 8	geoTriggerTempLowThreshold	Geoloc triggered on low temperature	Compact, Combo-compact
Bit 9	geoTriggerGeozoning	Geoloc stopped while in monitored area; enabled when leaving	Compact, Combo-compact

Geolocation technology object

A geolocation technology is an object that specifies the technology used and the associated action it performs.

Bit position	Flag name	Type	Description	Supported HW version
Bit 0–6	technology	String	Possible values: `NONE`, `LR11xx_A_GNSS`, `WIFI`, `BLE_SCAN1`, `BLE_SCAN2`, `AIDED_GNSS`, `GNSS`	Compact, Combo-compact
Bit 7	action	String	Possible values: `SKIP_ON_SUCCESS`, `ALWAYS_DONE`	Compact, Combo-compact

4.1.7.5 GNSS group This group configures the GNSS (MT3333).

group: “GNSS”			
Parameter name	Type	Description	Supported HW version
gnssConstellation	String	GNSS constellations to be used: `GPS_ONLY`, `GLONASS_ONLY`, `GPS_GLONASS`, `GPS_GALILEO`, `GPS_GLONASS_GALILEO`, `BEIDOU_ONLY`, `BEIDOU_GPS`	Compact, Combo-compact
gnssMaxTime	Number	GNSS max acquisition time	Compact, Combo-compact

group: “GNSS”			
gnssT0TimeoutStatic	Number	Max time to acquire at least one satellite when the tracker is static	Compact, Combo-compact
gnssEhpeStatic	Number	Expected estimated horizontal position error when the tracker is static	Compact, Combo-compact
gnssConvergenceStatic	Number	Extra-time after a first fix to refine the fix when the tracker is static	Compact, Combo-compact
gnssT0TimeoutMotion	Number	Max time to acquire at least one satellite when the tracker is in motion	Compact, Combo-compact
gnssEhpeMotion	Number	Expected estimated horizontal position error when the tracker is in motion	Compact, Combo-compact
gnssConvergenceMotion	Number	Extra-time after a first fix to refine the fix when the tracker is in motion	Compact, Combo-compact
gnssStandby	Number	Max time to let the device in standby mode	Compact, Combo-compact
gnssAgnssMaxTime	Number	Aided GNSS max acquisition time	Compact, Combo-compact
gnssT1Timeout	Number	Extra time in Aided GNSS mode to try doing a fix	Compact, Combo-compact

4.1.7.6 LR11xx group This group configures the LR11xx parameters.

group: “LR11xx”			
Parameter name	Type	Description	Supported HW version
lrConstellation	String	GNSS constellations to be used: `GPS_ONLY`, `BEIDOU_ONLY`, `BEIDOU_GPS`	Compact, Combo-compact
lrScanMode	String	Scan mode: `NAV1_SCAN`, `NAV2_SCAN`	Compact, Combo-compact
lrNbScans	Number	Number of scans for one position acquisition	Compact, Combo-compact
lrInterScanTime	Number	Time to wait between the scans for a position	Compact, Combo-compact
lrWifiReportNbBssid	Number	Max number of Wi-Fi BSSID per scan	Compact, Combo-compact
lrWifiMinNbBssid	Number	Minimum number of BSSID to consider the scan as success	Compact, Combo-compact

group: “LR11xx”			
lrWifiMinRssi	Number	Minimum RSSI to consider the BSSID. A null value disables the filter	Compact, Combo-compact
lrWifiBssidMacType	Number	MAC administration type of the BSSID to report	Compact, Combo-compact
lrRbeaconType	String	Recovery beacon type: `SHORT_ID`, `LONG_ID`	Compact, Combo-compact
lrRbeaconSosPeriod	Number	Recovery beacon period in SOS (seconds, 0–63)	Compact, Combo-compact
lrRbeaconMotionPeriod	Number	Recovery beacon period in motion (seconds, 0–63)	Compact, Combo-compact
lrRbeaconStaticPeriod	Number	Recovery beacon period in static (seconds, 0–63)	Compact, Combo-compact
lrRbeaconRangingTimeout	Number	Maximum timeout to exit the ranging mode (seconds, 120–300)	Compact, Combo-compact

4.1.7.7 BLE scan group There are 2 groups to configure two different BLE scans. Each scan has the same following parameters.

group: “BLE_SCAN1” / “BLE_SCAN2”			
Parameter name	Type	Description	Supported HW version
bleScanDuration	Number	Total time for a BLE scan in milliseconds	Compact, Combo-compact
bleScanWindow	Number	Scan window in milliseconds	Compact, Combo-compact
bleScanInterval	Number	Scan interval in milliseconds	Compact, Combo-compact
bleScanType	String	Type of beacons to scan: `ALL_BEACONS`, `EDDYSTONE_UUID`, `EDDYSTONE_URL`, `ALL_EDDYSTONE`, `IBEACON_ONLY`, `ALTBEACON_ONLY`, `CUSTOM`, `EXPOSURE_ADVERTISEMENT`	Compact, Combo-compact
bleScanMinRssi	Number	Min RSSI to consider the beacon, expressed in dB	Compact, Combo-compact

group: “BLE_SCAN1” / “BLE_SCAN2”			
bleScanMinNbBeacons	Number	Min number of beacons to consider the scan as success (solvable). Below this value the result will be not solvable	Compact, Combo-compact
bleScanFilter1Mask	Array of 10 bytes	Mask (10 bytes) to be applied to the ADV frame. Example: `{aa,11,14,FF,AE,00,00,00,00,00}`	Compact, Combo-compact
bleScanFilter1Value	Array of 10 bytes	Comparison value (10 bytes) belonging to filter 1	Compact, Combo-compact
bleScanFilter1Offset	Number	Offset in the ADV from which to apply filter 1	Compact, Combo-compact
bleScanFilter2Mask	Array of 10 bytes	Mask (10 bytes) to be applied to the ADV frame	Compact, Combo-compact
bleScanFilter2Value	Array of 10 bytes	Comparison value (10 bytes) belonging to filter 2	Compact, Combo-compact
bleScanFilter2Offset	Number	Offset in the ADV from which to apply filter 2	Compact, Combo-compact
bleScanNbBeacons	Number	Number of beacons to report (maximum 100)	Compact, Combo-compact
bleScanReportType	String	Scan report type: `MAC_ADDRESS`, `SHORT_IDENTIFIER`, `LONG_IDENTIFIER`	Compact, Combo-compact
bleScanReportIdOfs	Number	Offset in ADV to extract the beacon identifier	Compact, Combo-compact

4.1.7.8 Accelerometer group This group configures the accelerometer.

group: “ACCELEROMETER”			
Parameter name	Type	Description	Supported HW version
acceleroMotionSensi	Number	Motion sensitivity in mg	Compact, Combo-compact
acceleroMotionDuration	Number	Motion duration in seconds	Compact, Combo-compact
acceleroFullScale	String	Possible scale values: `2_G`, `4_G`, `8_G`, `16_G`	Compact, Combo-compact

group: “ACCELEROMETER”			
acceleroOutputDataRate	String	Output data rate: `12_5_HZ`, `25_HZ`, `50_HZ`, `100_HZ`, `200_HZ`	Compact, Combo-compact
acceleroShockThreshold	Number	Shock threshold in mg	Compact, Combo-compact

4.1.7.9 Network group This group configures the general networking.

group: “NETWORK”			
Parameter name	Type	Description	Supported HW version
netSelection	String	Define the networking type: `LORA_ONLY`, `CELLULAR_ONLY`, `LORA_FALLBACK_CELLULAR`, `CELLULAR_FALLBACK_LORA`	Compact, Combo-compact
netReconnectionSpacingStatic	Number	Time to wait before retrying to connect the main network when the device is static, in seconds	Compact, Combo-compact
netReconnectionSpacingMotion	Number	Time to wait before retrying to connect the main network when the device is in motion, in seconds	Compact, Combo-compact
netMainProbeTimeoutStatic	Number	Duration between each attempt to reconnect the main network when the device is static	Combo-compact
netMainProbeTimeoutMotion	Number	Duration between each attempt to reconnect the main network when the device is in motion	Combo-compact

4.1.7.10 LoRaWAN group This group configures the LoRaWAN networking.

group: “LORAWAN”			
Parameter name	Type	Description	Supported HW version
lorawanCnxTimeout	Number	Max time to wait for joining the network. The value 0 disables the timer	Compact, Combo-compact

group: “LORAWAN”			
lorawanHeartbeatPeriod	Number	Period at which a heartbeat message is sent to trigger a downlink. 0 disables the function	Compact, Combo-compact
lorawanProbeMaxAttemptsStatic	Number	Number of link-checks sent before declaring network as lost when the device is static	Compact, Combo-compact
lorawanProbeMaxAttemptsMotion	Number	Number of link-checks sent before declaring network as lost when the device is in motion	Compact, Combo-compact
lorawanProbePeriodStatic	Number	Time between link-check requests when the device is static	Compact, Combo-compact
lorawanProbePeriodMotion	Number	Time between link-check requests when the device is in motion	Compact, Combo-compact
lorawanConfirmNotifMap	Object	Map enabling the LoRaWAN confirmed message for notifications. Same byte layout as core notification enable object	Compact, Combo-compact
lorawanConfirmNotifRetry	Number	Number of retries for confirmed messages. Value 0: the number of TX follows ADR	Compact, Combo-compact
lorawanS1TxStrategy	Object	Socket 1. Transmission strategy. See transmission strategy object	Compact, Combo-compact
lorawanS1UlPort	Number	Socket 1. Uplink port	Compact, Combo-compact
lorawanS1DlPort	Number	Socket 1. Downlink port	Compact, Combo-compact

Transmission strategy object

The parameter `lorawanS1TxStrategy` is a bit-field on 3 bytes providing the LoRa transmission strategy.

Bit position	Flag Name	Type	Description	Supported HW version
Bit 0	ADREnabled	Boolean	Set to enable the LoRa network ADR if the tracker is static. Reset to disable the network ADR regardless the motion state of the tracker	Compact, Combo-compact
Bit 1	dualTransmissionInStaticEnabled	Boolean	Control the dual transmission when the tracker is static. Set to enable. Reset to disable	Compact, Combo-compact
Bit 2	dualTransmissionInMotionEnabled	Boolean	Control the dual transmission when the tracker is in motion. Set to enable. Reset to disable	Compact, Combo-compact
Bit 3	dataRateModification	Boolean	Control the DR (Datarate) modification for over-sized messages. Set to allow AOS to adapt the DR for messages not fitting the allowed size for a given datarate. Reset this flag to prevent sending of over-sized messages	Compact, Combo-compact
Bit 4–7	—	—	Reserved for future use (RFU)	

Bit position	Flag Name	Type	Description	Supported HW version
Bit 8–15	firstTransmissionDataRate	Number	Object to select each datarate for the first transmission. Example: `{"dr0": true, "dr1": false, "dr2": false, "dr3": true, "dr4": false, "dr5": false, "dr6": false, "dr7": true}`	Compact, Combo-compact
Bit 16–23	secondTransmissionDataRate	Number	Object to select each datarate for the second transmission	Compact, Combo-compact

4.1.7.11 Cellular group This group configures the cellular network.

group: "CELLULAR"			
Parameter name	Type	Description	Supported HW version
cellSimInterface	String	SIM interface: `SIM0`, `E_SIM`	Combo-compact
cellNetworkType	String	Cellular network type: `CELLULAR_NOT_USED`, `LTE_M`, `NB_IOT`	Combo-compact
cellSearchBands	Array of bytes	Radio frequency bands scanned to search a cell	Combo-compact
cellCnxTimeoutStatic	Number	Duration during which the modem searches for a cellular network when the tracker is static	Combo-compact
cellCnxTimeoutMotion	Number	Duration during which the modem searches for a cellular network when the tracker is in motion	Combo-compact
cellCnxNwReconnectTimeout	Number	Duration for the modem to automatically recover the network after a network is lost	Combo-compact
cellCnxMaxAttempts	Number	Number of times the network search is repeated before shutting down the modem	Combo-compact

group: “CELLULAR”			
cellAccessPointName	String	Service access point name. If not provided, retrieved from the SIM. Provide the value in ASCII format	Combo-compact
cellOperatorSimSlot0	String	Cellular operator name when using SIM0. Provide in ASCII format	Combo-compact
cellOperatorSimSlot1	String	Cellular operator name when using SIM1 (E.SIM). Provide in ASCII format	Combo-compact
cellLowPowerMode	String	Low power mode: `DISABLED`, `PSM`, `EDRX`, `PSM_EDRX`	Combo-compact
cellPsmTauPeriod	Object	Refer to cellular PSM TAU object	Combo-compact
cellPsmActiveTime	Object	Refer to cellular PSM active object	Combo-compact
cellEdrxPcl	Number	Requested paging cycle length	Combo-compact
cellEdrxPtw	Number	Requested paging time window	Combo-compact
cellS1TransportProto	String	Socket 1 transport protocol: `TCP`, `UDP`, `SECURE_TCP`, `SECURE_UDP`	Combo-compact
cellS1IpUrlAddr	String	Socket 1 remote IP address or URL (max 32 bytes). Provide in ASCII format	Combo-compact
cellS1DstIpPort	Number	Socket 1 destination UDP/TCP port	Combo-compact
cellS1SrcIpPort	Number	Socket 1 local UDP/TCP port number. Value 0 means the modem will choose one	Combo-compact
cellS1TxAggrTime	Number	Duration in seconds for which messages are held in the socket 1 transmit queue before being transmitted	Combo-compact
cellRestrictedPlmns	String	List of restricted Public Land Mobile Networks (PLMNs), ASCII format, max 32 bytes	Combo-compact

group: “CELLULAR”			
cellRaiTimeout	Number	RAI (Release Assistance Indication) timeout	Combo-compact
cellProbeMaxAttempts	Number	Number of echo-requests sent before declaring the network as lost	Combo-compact
cellProbePeriod	Number	Time between echo-requests	Combo-compact
cellApnUserId	String	User identifier for private APN	Combo-compact
cellApnUserPwd	String	User password for private APN	Combo-compact
cellApnAuthProtocol	String	Authentication protocol for private APN: `PAP`, `CHAP`	Combo-compact
cellFuotaServerIpUrlAddr	String	FUOTA server IP/URL in string format	Combo-compact
cellLpmRecoveryTimeout	Number	Timeout in seconds for the modem to automatically recover from low-power mode. Value 0 disables the recovery timer	Combo-compact

Cellular PSM TAU object

Bit position	Flag Name	Type	Description	Supported HW version
Bit 0–4	timerValue	Number	The timer value	
Bit 5–7	timerValueUnit	String	The unit: `VALUE_IS_INCREMENTED_IN_MULTIPLES_OF_10_MINU` `VALUE_IS_INCREMENTED_IN_MULTIPLES_OF_1_HOUR,` `VALUE_IS_INCREMENTED_IN_MULTIPLES_OF_10_HOUF` `VALUE_IS_INCREMENTED_IN_MULTIPLES_OF_2_SECON` `VALUE_IS_INCREMENTED_IN_MULTIPLES_OF_30_SECO` `VALUE_IS_INCREMENTED_IN_MULTIPLES_OF_1_MINUT` `VALUE_IS_INCREMENTED_IN_MULTIPLES_OF_320_HOU` `THE_TIMER_IS_DEACTIVATED`	Combo-compact

Cellular PSM active object

Bit position	Flag Name	Type	Description	Supported HW version
Bit 0–4	timerValue	Number	The timer value	

Bit position	Flag Name	Type	Description	Supported HW version
Bit 5–7	timerValueUnit	String	The unit: VALUE_IS_INCREMENTED_IN_MULTIPLES_OF_2_SECON VALUE_IS_INCREMENTED_IN_MULTIPLES_OF_1_MINUT VALUE_IS_INCREMENTED_IN_MULTIPLES_OF_1_DECI_ THE_TIMER_IS_DEACTIVATED	Combo-compact

4.1.7.12 BLE group This group configures the BLE connectivity and beaconing parameters.

group: "BLE"			
Parameter name	Type	Description	Supported HW version
bleCnxTxPower	Number	BLE Tx power level	Compact, Combo-compact
bleCnxAdvDuration	Number	Time to wait before stopping advertising or switching to slow advertising	Compact, Combo-compact
bleCnxBehavior	String	BLE connectivity configuration: DISABLE, ENABLE_NO_PASSKEY, ENABLE_PASSKEY, ENABLE_NO_PASSKEY_NO_SLOW_ADV, ENABLE_PASSKEY_NO_SLOW_ADV	Compact, Combo-compact
bleBeaconTxPower	Number	BLE beacon Tx power level	Compact, Combo-compact
bleBeaconType	String	BLE beacon type: DISABLE, EDDYSTONE_UID, IBEACON, ALTBEACON, QUUPPA, EXPOSURE_ADVERTISEMENT	Compact, Combo-compact
bleBeaconIdentifier	Array	BLE beaconing ID parameter	Compact, Combo-compact
bleBeaconFastAdvInterval	Number	BLE beacon fast advertising interval in ms (minimum 40)	Compact, Combo-compact
bleBeaconSlowAdvInterval	Number	BLE beacon slow advertising interval in ms (minimum 40)	Compact, Combo-compact
bleScanPresencePeriod	Number	Period in seconds at which the presence detection BLE scan is triggered (0 disables). Range: 0–86400	Compact, Combo-compact

group: “BLE”			
bleScanPresenceMaxTime	Number	Maximum time in seconds a device is considered present after last detection (0 disables). Range: 0–86400	Compact, Combo-compact

4.1.7.13 Telemetry group This group configures the telemetry measurement feature. Parameters are organized per sensor (sensor0–sensor3). On the Charp Maxi (ABW010W): sensor0 = Light, sensor1 = Temperature, sensor2 = Relative Humidity, sensor3 = not used. Sensors 1 and 2 are sampled simultaneously using `sensor1MeasNsampling`.

group: “TELEMETRY”			
Parameter name	Type	Description	Supported HW version
nmetaDataperiod	Number	Periodicity of metadata frame transmission in seconds. When set to 0, payload length is limited to 51 bytes (metadata disabled)	Combo-compact
sensor0MeasNsampling	Number	Sampling period for sensor 0 (Light) in seconds. Set to 0 to disable	Combo-compact
sensor0HighThreshold	Number	High alarm threshold for sensor 0. Triggers immediate transmission of the timeseries buffer when value exceeds this limit	Combo-compact
sensor0LowThreshold	Number	Low alarm threshold for sensor 0. Triggers immediate transmission when value drops below this limit	Combo-compact
sensor0Hysteresis	Number	Hysteresis to prevent successive alarms for sensor 0	Combo-compact
sensor0MeasNmaxinterval	Number	Maximum time between two consecutive records for sensor 0, in seconds	Combo-compact
sensor0TelemNmaxinterval	Number	Maximum time between two timeseries transmissions for sensor 0, in seconds	Combo-compact

group: “TELEMETRY”			
sensor0CyclicVersion	Number	Cyclic version tag for sensor 0 (3-bit, 0–7). Increment after configuration changes to signal the codec that parameters have changed	Combo-compact
sensor1MeasNsampling	Number	Sampling period for sensor 1 (Temperature) in seconds. Also controls sensor 2 (Humidity) since both sensors are sampled simultaneously	Combo-compact
sensor1HighThreshold	Number	High alarm threshold for sensor 1 (Temperature)	Combo-compact
sensor1LowThreshold	Number	Low alarm threshold for sensor 1 (Temperature)	Combo-compact
sensor1Hysteresis	Number	Hysteresis for sensor 1 alarm	Combo-compact
sensor1MeasNmaxinterval	Number	Maximum time between two consecutive records for sensor 1, in seconds	Combo-compact
sensor1TelemNmaxinterval	Number	Maximum time between two timeseries transmissions for sensor 1, in seconds	Combo-compact
sensor1CyclicVersion	Number	Cyclic version tag for sensor 1 (3-bit, 0–7)	Combo-compact
sensor2MeasNsampling	Number	Sampling period for sensor 2 (Relative Humidity) in seconds. Set to 0 to disable (sensor 1 controls both sensors)	Combo-compact
sensor2HighThreshold	Number	High alarm threshold for sensor 2 (Humidity)	Combo-compact
sensor2LowThreshold	Number	Low alarm threshold for sensor 2 (Humidity)	Combo-compact
sensor2Hysteresis	Number	Hysteresis for sensor 2 alarm	Combo-compact
sensor2MeasNmaxinterval	Number	Maximum time between two consecutive records for sensor 2, in seconds	Combo-compact

group: “TELEMETRY”			
sensor2TelemNmaxinterval	Number	Maximum time between two timeseries transmissions for sensor 2, in seconds	Combo-compact
sensor2CyclicVersion	Number	Cyclic version tag for sensor 2 (3-bit, 0–7)	Combo-compact
sensor3MeasNsampling	Number	Sampling period for sensor 3 in seconds (not used in Charp Maxi)	Combo-compact
sensor3HighThreshold	Number	High alarm threshold for sensor 3	Combo-compact
sensor3LowThreshold	Number	Low alarm threshold for sensor 3	Combo-compact
sensor3Hysteresis	Number	Hysteresis for sensor 3 alarm	Combo-compact
sensor3MeasNmaxinterval	Number	Maximum time between two consecutive records for sensor 3, in seconds	Combo-compact
sensor3TelemNmaxinterval	Number	Maximum time between two timeseries transmissions for sensor 3, in seconds	Combo-compact
sensor3CyclicVersion	Number	Cyclic version tag for sensor 3 (3-bit, 0–7)	Combo-compact

4.1.8 Telemetry uplink object The decoder output includes the `telemetry` object when `header.type === "TELEMETRY"`. These uplinks are transmitted on fPort 19 and originate from the telemetry recording engine of the Charp Maxi device (ABW010W). They are independent from NOTIFICATION uplinks and carry sensor data in two sub-formats identified by the `telemetry.type` field.

`telemetry.type`	Description
`"metadata"`	Configuration frame describing the active TelemetryIDs and their measurement parameters
`"timeseries"`	Data frame carrying timestamped measurement samples for a single TelemetryID

4.1.8.1 Metadata payload A metadata payload is transmitted periodically (period set by `nmetaDataperiod`) and whenever the device configuration changes. It must be received and stored by the application to correctly decode subsequent timeseries payloads (see Section 4.1.8.5 — Context array).

The `telemetry` object for a metadata payload:

Parameter	Type	Description
type	String	Always `"metadata"`

Parameter	Type	Description
TelemetryIDs	Object	Object keyed by telemetryId (`"0"`, `"1"`, `"2"` …). Each value describes one sensor

Each entry inside `TelemetryIDs`:

Parameter	Type	Description
telemetryId	Number	Sensor identifier
cyclicVersion	Number	3-bit version tag. Changes when the measurement configuration is updated
fixedTimeInterval	Number	2-bit flag encoding the interval mode. `0` = fixed interval. Value transmitted in the metadata frame
measurementNMaxInterval	Number	Maximum time in seconds between two consecutive records
telemetryIDMaxInterval	Number	Maximum time in seconds between two transmissions of this TelemetryID
numMeasurements	Number	Number of MeasurementNIDs per timeseries (always 1 in Phase 1)
timestamp	String	ISO 8601 UTC timestamp when this metadata frame was generated
measurementConfig	Object	See Section 4.1.8.3 — measurementConfig object

Example metadata payload output:


```
{
  "header": { "type": "TELEMETRY", ... },
  "telemetry": {
    "type": "metadata",
    "TelemetryIDs": {
      "0": {
        "telemetryId": 0,
        "cyclicVersion": 0,
        "fixedTimeInterval": 0,
        "measurementNMaxInterval": 600,
        "telemetryIDMaxInterval": 12000,
        "numMeasurements": 1,
        "timestamp": "2025-05-21T10:04:36.000Z",
        "measurementConfig": {
          "measurementId": 0,
          "dataType": "16_BIT_SIGNED",
```

```
          "recordingPolicy": "INSTANT",
          "codingPolicy": "DELTA_COMPRESSION",
          "sampleCompression": "LINEAR",
          "scalingFactor": 1
        }
      },
      "1": {
        "telemetryId": 1,
        "cyclicVersion": 1,
        "measurementConfig": { "scalingFactor": 0.1, ... }
      },
      "2": {
        "telemetryId": 2,
        "cyclicVersion": 2,
        "measurementConfig": { "scalingFactor": 0.1, ... }
      }
    }
  }
}
```

4.1.8.2 Timeseries payload A timeseries payload carries the buffered measurement samples for a single TelemetryID. The codec requires a valid metadata context to decode it (see Section 4.1.8.5).

Parameter	Type	Description
type	String	Always `"timeseries"`
telemetryId	Number	Sensor identifier
cyclicVersion	Number	Must match the `cyclicVersion` from the last received metadata for this `telemetryId`. A mismatch means the configuration has changed
cyclicCounter	Number	5-bit rolling frame counter (0–31). Used to detect missing frames
alarmTrigger	Boolean	`true` when transmission was triggered by a threshold alarm instead of a periodic interval
measurementConfig	Object	Coding parameters (inherited from last metadata). See Section 4.1.8.3
measurements	Array	Timestamped raw measurement samples, ordered from **most recent to oldest**. See Section 4.1.8.4

Example timeseries payload output (telemetryId=0, Light sensor):

```
{
  "header": { "type": "TELEMETRY", "batteryLevel": 100, ... },
```

```
  "telemetry": {
    "type": "timeseries",
    "telemetryId": 0,
    "cyclicVersion": 0,
    "cyclicCounter": 6,
    "alarmTrigger": false,
    "measurementConfig": {
      "measurementId": 0,
      "dataType": "16_BIT_SIGNED",
      "recordingPolicy": "INSTANT",
      "codingPolicy": "DELTA_COMPRESSION",
      "sampleCompression": "LINEAR",
      "scalingFactor": 1
    },
    "measurements": [
      { "timestamp": "2025-05-23T08:34:08.000Z", "value": 1660 },
      { "timestamp": "2025-05-23T08:24:08.000Z", "value": 1642 },
      ...
      { "timestamp": "2025-05-23T05:14:08.000Z", "value": 665 }
    ]
  }
}
```

4.1.8.3 measurementConfig object

Parameter	Type	Description
measurementId	Number	Measurement sub-identifier within a TelemetryID (2-bit field, range 0–3). In Phase 1 firmware, `numMeasurements` is always 1, so this value is always `0`. The field is reserved for future firmware phases that may report multiple measurements per `telemetryId`. Must not be confused with `telemetryId` which identifies the physical sensor
dataType	String	Raw data encoding: `8_BIT_UNSIGNED` or `16_BIT_SIGNED`
recordingPolicy	String	How the value is recorded at each interval: `INSTANT` (value at recording time), `MIN` (minimum since last interval), `MAX` (maximum since last interval), `CHANGE_OF_VALUE`
codingPolicy	String	Payload compression algorithm: `NO_COMPRESSION` or `DELTA_COMPRESSION`

Parameter	Type	Description
sampleCompression	String	Interpolation curve between samples: `LINEAR` or `LOGARITHMIC`
scalingFactor	Number	Multiplier applied internally by the codec to convert raw values to physical units. Provided for reference only — the `value` field in `measurements` is already in physical units (see Section 4.1.8.4)

4.1.8.4 Measurements array The `measurements` array contains timestamped samples ordered from **most recent** (index 0) to **oldest** (last index).

Parameter	Type	Description
timestamp	String	ISO 8601 UTC timestamp of the measurement
value	Number	Physical measurement value — the `scalingFactor` has already been applied by the codec

The codec applies `scalingFactor` internally before returning the decoded output, so no further calculation is needed:

```
// codec internally computes:
value = parseFloat((rawValue * scalingFactor).toFixed(2))
```

Examples for Charp Maxi (values are ready to use as-is):

> **Note:** `scalingFactor` is not fixed — it is transmitted in the metadata frame and may vary depending on firmware configuration. The examples below reflect the values reported in the Charp Maxi Phase 1 metadata.

telemetryId	Sensor	Decoded value	Interpretation
0	Light	665	665 lux
1	Temperature	24.6	24.6 °C
2	Relative Humidity	65.0	65.0 %RH

4.1.8.5 Context array Timeseries payloads are delta-compressed using parameters defined in the metadata payload. Since metadata and timeseries arrive in separate uplinks, the codec relies on a **persistent context** to carry the metadata configuration across calls.

How context works:

- When the codec decodes a metadata payload, it writes the decoded `TelemetryIDs` configuration into context.
- When the codec decodes a timeseries payload, it reads from context to retrieve the corresponding decoding parameters.

- Without valid metadata in context, timeseries decoding fails with:

```
{ "errors": ["Context is empty, cannot retrieve metadata history"] }
```

Application responsibility — context lifecycle:

The driver exports `isContextUsedInPayload(input)` to help the calling application manage context:

1. Before calling `decodeUplink`, the application calls `isContextUsedInPayload(input)`. If it returns `true`, the payload is a telemetry frame and context management is required.
2. The application loads the previously stored context (from its database or memory) into the global `context` variable (`global.context` in Node.js / `window.context` in browser) before calling `decodeUplink`.
3. After decoding a metadata payload, the codec updates the global `context`. The application must save this updated context (e.g. per DevEUI) for use with subsequent timeseries payloads.

The `context` field in `examples.json` (`input.context`) follows this same contract — the test runner reads `input.context` and loads it into the global `context` variable before invoking `decodeUplink`.

The context must be refreshed whenever a new metadata payload is received — a changed `cyclicVersion` on any TelemetryID signals that the device configuration has been updated.

---

5. Decoding multiple payloads

Some uplink messages may span multiple LoRaWAN frames (multi-frame uplinks). When the `multiFrame` flag is set to `true` in the Header object, the uplink belongs to a group of frames that must be reassembled.

The Extended Header object provides the information needed to reassemble multi-frame uplinks: - `groupId` identifies which group the frame belongs to. - `frameNumber` gives the position of this frame within the group (0 = first frame). - `last` signals whether this is the last frame in the group.

Multi-frame uplinks are used for large payloads such as almanac data and status pages. The caller application must buffer frames belonging to the same `groupId` and reassemble them in order of `frameNumber`, then call `decodeUplink` on the assembled payload.

On cellular networks (LTE), each uplink includes a 10-byte cellular network header prepended before the application payload:

Bytes	Field	Description
0–7	DevEUI	Device EUI (big-endian)
8–9	Frame counter	Uplink frame counter (MSB, LSB)

The codec automatically detects and strips this header when present.

---

6. Encoding a downlink

The encoder takes a JSON format object and converts it into the byte array for downlink transmission.

Each downlink message must include the following fields: - `downMessageType`: Type of downlink messages (string format). Acceptable values are: - `COMMAND` - `REQUEST` - `ANSWER` - `ackToken`: A random integer ranging from 0 to 7 used by the tracker to acknowledge the downlink message.

Notes: - Each field name must be encoded as a string. - Additional fields must be provided based on the type of downlink message.

6.1 Commands

This message sends a command to the tracker. An additional field, `commandType`, must be included with one of the following string values:

- `CLEAR_AND_RESET` — Clear configuration in flash (with optional parameter preservation list), then reset
- `RESET` — Reset the device
- `START_SOS` — Activate SOS mode
- `STOP_SOS` — Deactivate SOS mode
- `SYSTEM_STATUS_REQUEST` — Request a status notification uplink
- `POSITION_ON_DEMAND` — Request an immediate position fix
- `SET_GPS_ALMANAC` — Upload GPS almanac data to the device
- `SET_BEIDOU_ALMANAC` — Upload BeiDou almanac data to the device
- `START_BLE_CONNECTIVITY` — Start BLE advertising
- `STOP_BLE_CONNECTIVITY` — Stop BLE advertising
- `SYSTEM_EVENT` — Trigger a virtual system event
- `CLEAR_MOTION_PERCENTAGE` — Reset the motion percentage counter
- `GET_DATA_BUFFERING_ENTRIES` — Request buffered data replay
- `CLEAR_ALL_DATA_BUFFERING` — Delete all buffered data
- `CLEAR_BLE_BOND_DATA` — Remove BLE bonding information

The `commandType` specifies the type of command being sent. Note: Commands do not expect responses or statuses from the tracker, except for `SYSTEM_STATUS_REQUEST` and `POSITION_ON_DEMAND` which trigger notification/position uplinks.

Additional fields for SYSTEM_EVENT:

```
{
  "downMessageType": "COMMAND",
  "commandType": "SYSTEM_EVENT",
  "ackToken": 0,
  "classId": "CORE",
  "eventType": 5
}
```

`classId` is the Event Manager class and must be one of: `BUTTON_1`, `BUTTON_2`, `BUZZER`, `ACCELEROMETER`, `POWER`, `TEMPERATURE`, `GEOLOCATION`, `CONFIGURATION`, `NETWORK`, `CORE`, `BLE`, `USER`, `FUOTA`. Per the firmware reference guide, this command is intended to trigger events belonging to the `CORE` and `USER` classes only, though the wire format allows any of the 13 classes.

`eventType` is a raw numeric byte (0–255) whose meaning depends on the class and is defined by the firmware specification. The codec transmits/decodes the value without translating it to a named event type.

Additional fields for GET_DATA_BUFFERING_ENTRIES:

Field	Type	Description
beginUtcTime	String (ISO 8601)	Start of the data replay window as a UTC timestamp

Field	Type	Description
duration	Number	Duration of the replay window in seconds (0–65535)
bufferedDataType	Object	Boolean sub-fields: `position` (bit 0), `notification` (bit 1), `telemetry` (bit 2)

```
{
  "downMessageType": "COMMAND",
  "commandType": "GET_DATA_BUFFERING_ENTRIES",
  "ackToken": 0,
  "beginUtcTime": "2025-06-01T00:00:00.000Z",
  "duration": 3600,
  "bufferedDataType": {
    "position": true,
    "notification": true,
    "telemetry": false
  }
}
```

6.2 Requests

This message sends a request to the tracker. An additional field, `requestType`, must be included with one of the following string values: - `GENERIC_CONFIGURATION_SET` - `PARAM_CLASS_CONFIGURATION_SET` - `GENERIC_CONFIGURATION_GET` - `PARAM_CLASS_CONFIGURATION_GET` - `BLE_STATUS_CONNECTIVITY` - `CRC_CONFIGURATION_REQUEST` - `SENSOR_REQUEST` - `DEBUG_INFO_REQUEST` - `FUOTA_REQUEST` - `RECOVERY_BEACON_KEY_UPDATE` - `RECOVERY_BEACON_ROOT_KEY_INDEX_GET`

Additional fields must be provided based on the type of request. Note: Requests expect a response uplink from the tracker.

6.2.1 Generic configuration set request This request enables the configuration of any tracker parameter. It should be used when configuring multiple parameters from different classes (groups). The `requestType` must be `GENERIC_CONFIGURATION_SET`.

The `setGenericParameters` field is mandatory for this request. It is an array of objects where each object represents a group and its parameters.

Parameter Name	Type	Description	Supported HW version
group	String	Refer to group	Compact, Combo-compact
parameters	Array of object	Each object has two attributes: `parameterName` (see parameter configuration) and `parameterValue` (the value to set)	Compact, Combo-compact

Example:

```
{
  "downMessageType": "REQUEST",
  "requestType": "GENERIC_CONFIGURATION_SET",
  "ackToken": 1,
  "setGenericParameters": [
    {
      "group": "SYSTEM_CORE",
      "parameters": [
        { "parameterName": "coreMonitoringPeriod", "parameterValue": 300 }
      ]
    },
    {
      "group": "GEOLOC",
      "parameters": [
        { "parameterName": "geolocMotionPeriod", "parameterValue": 120 }
      ]
    }
  ]
}
```

6.2.2 Parameter class configuration set request This request enables the configuration of parameters in a single parameter class in the tracker. Multiple parameters within this class can be modified. The `requestType` must be `PARAM_CLASS_CONFIGURATION_SET`.

The `setParameterClass` field is mandatory for this request. It is an object that represents a group and its parameters.

Parameter Name	Type	Description	Supported HW version
group	String	Refer to group	Compact, Combo-compact
parameters	Array of object	Each object has two attributes: `parameterName` and `parameterValue`	Compact, Combo-compact

Example:

```
{
  "downMessageType": "REQUEST",
  "requestType": "PARAM_CLASS_CONFIGURATION_SET",
  "ackToken": 2,
  "setParameterClass": {
    "group": "LORAWAN",
    "parameters": [
      { "parameterName": "lorawanS1UlPort", "parameterValue": 18 }
    ]
  }
}
```

6.2.3 Generic class configuration get request This request retrieves specific configuration parameters from the tracker from multiple classes (groups). The `requestType` must be `GENERIC_CONFIGURATION_GET`.

The `getGenericParameters` field is mandatory. It is an array of objects where each object represents a group and its parameters.

Parameter Name	Type	Description	Supported HW version
group	String	Refer to group	Compact, Combo-compact
parameters	Array of string	An array of strings, each representing a parameter name within the same group. Each string must have a valid parameter name as specified in the parameter configuration	Compact, Combo-compact

Example:

```
{
  "downMessageType": "REQUEST",
  "requestType": "GENERIC_CONFIGURATION_GET",
  "ackToken": 3,
  "getGenericParameters": [
    {
      "group": "SYSTEM_CORE",
      "parameters": ["coreMonitoringPeriod", "coreStatusPeriod"]
    }
  ]
}
```

6.2.4 Parameter class configuration get request This request retrieves specific configuration parameters from a single parameter class in the tracker. The `requestType` must be `PARAM_CLASS_CONFIGURATION_GET`.

The `getParameterClass` field is mandatory. It is an object that represents a group and its parameters.

Parameter Name	Type	Description	Supported HW version
group	String	Refer to group	Compact, Combo-compact

Parameter Name	Type	Description	Supported HW version
parameters	Array of string	An array of strings, each representing a parameter name within the same group. Each string must have a valid parameter name as specified in the parameter configuration	Compact, Combo-compact

Example:

```
{
  "downMessageType": "REQUEST",
  "requestType": "PARAM_CLASS_CONFIGURATION_GET",
  "ackToken": 4,
  "getParameterClass": {
    "group": "CELLULAR",
    "parameters": ["cellNetworkType", "cellSimInterface"]
  }
}
```

6.2.5 BLE connectivity status request This request queries the BLE connectivity status of the device. The `requestType` must be `BLE_STATUS_CONNECTIVITY`. No additional fields are required.

Example:

```
{
  "downMessageType": "REQUEST",
  "requestType": "BLE_STATUS_CONNECTIVITY",
  "ackToken": 4
}
```

6.2.6 CRC configuration request This request asks the device to return the CRC(s) of its configuration. The `requestType` must be `CRC_CONFIGURATION_REQUEST`.

The `crc` field is an array of group name strings. If the array contains all group names, a global CRC is requested.

Parameter Name	Type	Description	Supported HW version
crc	Array of String	Non-empty list of group names to request CRC for. See group. Must contain at least one group name — an empty array is rejected by the encoder. To request a global CRC, include all group names	Compact, Combo-compact

Example — request per-group CRCs:

```
{
  "downMessageType": "REQUEST",
  "requestType": "CRC_CONFIGURATION_REQUEST",
  "ackToken": 5,
  "crc": ["SYSTEM_CORE", "GEOLOC", "LORAWAN"]
}
```

6.2.7 Get sensor values request This request asks the device to return current sensor readings. The `requestType` must be `SENSOR_REQUEST`.

The `sensorIds` field is an array of sensor objects.

Parameter Name	Type	Description	Supported HW version
sensorIds	Array of object	Each object has `type` attribute: `ACCELEROMETER` (id=1). Note: sensor type `DO_NOT_USE` (id=0) is accepted by the encoder but is reserved and must not be sent	Compact, Combo-compact

Example:

```
{
  "downMessageType": "REQUEST",
  "requestType": "SENSOR_REQUEST",
  "ackToken": 6,
  "sensorIds": [{ "type": "ACCELEROMETER" }]
}
```

6.2.8 Get debug information request This request asks the device to return debug or crash information. The `requestType` must be `DEBUG_INFO_REQUEST`.

Parameter Name	Type	Description	Supported HW version
debugInfoType	Number	Type of debug info requested: `0` = crash info	Compact, Combo-compact

Example:

```
{
  "downMessageType": "REQUEST",
  "requestType": "DEBUG_INFO_REQUEST",
  "ackToken": 7,
  "debugInfoType": 0
}
```

6.2.9 FUOTA request This request initiates a firmware update over the air. The `requestType` must be `FUOTA_REQUEST`.

Parameter Name	Type	Description	Supported HW version
fuotaBinaryType	String	The firmware binary to update: `APPLICATION`, `BLE_STACK`, `MT3333`, `LR11XX`	Combo-compact
filePath	String	File path on the FUOTA server. Null-terminated ASCII string, max 46 characters	Combo-compact

Example:

```
{
  "downMessageType": "REQUEST",
  "requestType": "FUOTA_REQUEST",
  "ackToken": 0,
  "fuotaBinaryType": "APPLICATION",
  "filePath": "/firmware/at3-v1.6-0.bin"
}
```

6.2.10 Recovery beacon key update request This request updates the recovery beacon cryptographic keys on the device. The `requestType` must be `RECOVERY_BEACON_KEY_UPDATE`.

Parameter Name	Type	Description	Supported HW version
k0Key	String	Root key K0, 16 bytes as a hex string (32 hex characters)	Compact, Combo-compact
k1Key	String	Root key K1, 16 bytes as a hex string (32 hex characters)	Compact, Combo-compact
rootKeyIndex	Number	Root key index (0–7)	Compact, Combo-compact

Example:

```
{
  "downMessageType": "REQUEST",
  "requestType": "RECOVERY_BEACON_KEY_UPDATE",
  "ackToken": 1,
  "k0Key": "000102030405060708090a0b0c0d0e0f",
  "k1Key": "101112131415161718191a1b1c1d1e1f",
  "rootKeyIndex": 0
}
```

6.2.11 Recovery beacon root key index get request This request retrieves the current root key index used by the recovery beacon. The `requestType` must be `RECOVERY_BEACON_ROOT_KEY_INDEX_GET`. No additional fields are required.

Example:

```
{
  "downMessageType": "REQUEST",
  "requestType": "RECOVERY_BEACON_ROOT_KEY_INDEX_GET",
  "ackToken": 2
}
```

6.3 Answers

Answers are downlinks sent by the server in response to uplink Query messages. The `downMessageType` must be `ANSWER`.

Answer types:

answerType	Answers query	Description
`AIDING_POSITION`	Query type 0	Provides aiding position data to the device
`ECHO_REPLY`	Query type 1	Echo response (same sequence number as the request)
`UPDATE_GPS_ALMANAC`	Query type 2	Provides GPS almanac data for requested satellites
`UPDATE_BEIDOU_ALMANAC`	Query type 3	Provides BeiDou almanac data for requested satellites

Example — Echo reply:

```
{
  "downMessageType": "ANSWER",
  "answerType": "ECHO_REPLY",
  "ackToken": 0,
  "sequenceNumber": 42
}
```

---

7. Decoding a downlink

The codec can decode LoRaWAN downlink payloads sent to an Abeeway Tracker running AT3 firmware.

The output from the decoder is a JSON-formatted object. Each JSON output includes the following fields: - `downMessageType`: Type of downlink message (string format). Acceptable values are: - `COMMAND` - `REQUEST` - `ANSWER` - `ackToken`: The acknowledgement token (0–7).

Depending on the downlink type, the fields in the JSON output will vary, matching those described for encoding a downlink.

Note: for `downMessageType ANSWER`, only `answerType` is decoded (see Answers for the possible values). The answer's data part (e.g. `sequenceNumber` for `ECHO_REPLY`) is not decoded.

Example — decoding an Answer:

Input bytes: `[0x18, 0x01]`

Output:

```
{
  "downMessageType": "ANSWER",
  "ackToken": 0,
  "answerType": "ECHO_REPLY"
}
```

Example — decoding a RESET command:

Input bytes: `[0x09, 0x01]`

Output:

```
{
  "downMessageType": "COMMAND",
  "commandType": "RESET",
  "ackToken": 1
}
```

Example — decoding a GET_DATA_BUFFERING_ENTRIES command:

```
{
  "downMessageType": "COMMAND",
  "commandType": "GET_DATA_BUFFERING_ENTRIES",
  "ackToken": 0,
  "beginUtcTime": "2025-06-01T00:00:00.000Z",
  "duration": 3600,
  "bufferedDataType": {
    "position": true,
    "notification": true,
    "telemetry": false
  }
}
```