Codec Asset Tracker v1.7

V1.3

12/03/2019

ABEEWAY SAS
www.abeeway.com – contact@abeeway.com

NOTICE

REVISIONS

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Hend Affes	Initial version	06/09/2018
1.1	Patrick Beatini	Rework	10/09/2018
1.2	Hend Affes	Update in line with AT 1.7.3	25/02/2019
1.3	PY Gueniffey	Added downlink encoder snippet	12/03/2019

Table des matières

1 Introduction

This document describes the LoRa codec (COder/DECoder) for the Abeeway Asset Tracker 2 firmware version V1.7. It explains how to decode the uplinks and encode the downlinks messages.

For a full description of the Abeeway trackers, It is advised to read the companion document: abeeway-reference-guide-fw1.7.pdf.

The document provides the installation information and the codec use. It also provides examples. To separate the examples from the rest of the documents, they are always enclosed by the following tags:

---------- Snippet begin ----------
---------- Snippet end ----------

The codec uses the Java language and comes in a form of a compressed tarball (jar file).

Once the decoder has been installed, the application using it should first create the driver using the following statement:

---------- Snippet begin ----------
```
private static DriverEngine driverEngine = new DriverEngine().loadAll();
```
---------- Snippet end ----------

2. Installation

To use the codec, you should perform the following instructions:

1. Download the iot-flow-assettracker-driver-0.9-jar-with-dependencies.jar
2. Create your Maven project
3. Add the jar file in your lib folder
4. Add the following dependencies in your pom.xml file:

---------- Snippet begin ----------

```
<dependencies>
    ...
    <!-- Driver -->
    <dependency>
        <groupId>com.actility.iot.flow</groupId>
        <artifactId>iot-flow-assettracker-driver</artifactId>
        <version>0.9</version>
        <scope>system</scope>

<systemPath>${basedir}/lib/iot-flow-assettracker-driver-0.9-jar-with-dependencies.jar</
systemPath>
    </dependency>
    <!-- Logging -->
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-simple</artifactId>
        <version>1.7.21</version>
    </dependency>
    ...
</dependencies>
```

---------- Snippet end ----------

3 Decoding an uplink

Description

The input of the decoder is a LoRa uplink payload coming from an Abeeway Tracker using a firmware Asset Tracker version 1.7.
The output of the decoder is a string in the well known JSON format.
For example, a LoRa heartbeat message contains only the following fields:

- Moving
- On-demand
- Battery level
- Temperature
- Ack token.
- Reset cause
- Firmware version

The input parameters are described in the table below. Note the **Msg** column refers to the LoRa message type and **Reference** gives the Byte/bits position in the LoRa payload. The companion document abeeway-reference-guide-fw1.7.pdf should be used for more details.
Example:
The **Reference** uses the following convention:

- Condition: Equal sign means a condition. Example *B4[3-0]=0* : bits[0-3] of the byte 4 should be equal to 0.
- Condition multiple: Equals sign followed by a bracket. Example B4[3-0]={0,2} means that the bits[0-3] of the byte 4 should be equal to 0 or 2.
- Single symbol means a value coded on 8 bits. Example B0: value is given by the byte 0.
- Value array uses a minus sign separator. Example: *B6-B9*: the value is given by the bytes 6 to 9.

Parameter	Type	Description	Msg	Reference
latitude	Number	Latitude of the position (expressed in degrees).	0x3	B4[3-0]=0 B6-B9
longitude	Number	Longitude of the position (expressed in degrees).	0x3	B4[3-0]=0 B9-B11
EHPE	Number	Estimated Horizontal Position Error, a measure of the error in a GPS position in the horizontal plane It is expressed in meters	0x3	B4[3-0]=0 B12
messageType	String	Indicates the uplink type: ◦ MESSAGE_POSITION ◦ HEARTBEAT ◦ ENERGY_STATUS ◦ SHUTDOWN ◦ FRAME_PENDING ◦ DEBUG ◦ MISC_DATA ◦ EVENT ◦ UNKNOWN	All	B0

fixAge	Number	Age in seconds of message position given by the GPS technology only.	0x3	B4[3-0]=0 B5
mode	String	Tracker operating mode. Values can be: ◦ STANDBY ◦ OFF ◦ MOTION ◦ PERMANENT ◦ START_END ◦ ACTIVITY ◦ UNKNOWN	All	B1[5-7]
batteryLevel	Number	The battery voltage. is expressed in Volts If the value is equal to zero, it is mean that the battery is charging	All	B2
ackToken	Number	Acknowledgement token is used to inform the server of the reception of LoRa downlink messages.	All	B4[7-4]
firmwareVersion	String	Firmware Version	0x5	B6-8
resetCause	String	Reset cause	0x5	B5
fixType	String	Fix type describes what kind of data is in the uplink. Values can be: ◦ GPS_FIX ◦ GPS_TIMEOUT ◦ WIFI_BSSID ◦ WIFI_TIMEOUT ◦ WIFI_FAILURE ◦ XGPS_DATA ◦ XGPS_DATA_WITH_GPS_SW_TIME ◦ BLE_BEACON_SCAN ◦ BLE_BEACON_FAILURE ◦ WIFI_BSSIDS_WITH_NO_CYPHER ◦ UNKNOWN	0x3	B4[3-0]
periodicPosition	Boolean	True if this is a periodic position message.	0x3	B1[0]
gpsOnRuntime	Number	The total time in seconds spent by the GPS in state ON since boot. It is transmitted in energy status message	0x4	B5-8
gpsStandbyRuntime	Number	The total time in seconds spent by the GPS in state STANDBY since boot. It is transmitted in energy status message	0x4	B9-12
wifiScanCount	Number	The total counter of WIFI scans since boot. No unit (counter). It is transmitted in energy status message	0x4	B13-16
timeoutCause	String	The GPS timeout cause. Values can be • USER-TIMEOUT: the GPS was not able to compute a fix before the position message period, • UNKNOWN	0x3	B4[3-0]=1 B5

bestSatellitesIndicator	Array	when a "GPS Timeout" occurs, the rest of the LoRa message consists of the type of timeout that preempted the scan and the 4 best satellites C/N indicators The bestSatellitesIndicator is an array.	0x3	B4[3-0]=1 B6-B9
temperature	Number	Board temperature in °C	All	B2
miscDataTag	String	Values can be : • ACTIVITY_COUNTER • DEVICE_CONFIGURATION • UNKNOWN	0x7	B5
userAction	Number	Set if the user alert/SOS has been entered.	All	B1[5]
appState	Number	Tracking state. 1: tracking. 0: idle	All	B1[3]
moving	Boolean	Moving. True if the tracker is currently moving.	All	B1[2]
onDemand	Boolean	True if this is an on-demand message.	All	B1[0]
VoltageMeasures	Array	Voltage measures reported under WIFI scan failures.	0x3	B4[3-0]=3 B5-B10
debugCommandTag	String	Values can be : • DEBUG_CRASH_INFORMATIO N • TX_POWER_INDEX_VALUE • UNKNOWN	0xF	B4[3-0]
errorCode	Number	Error code.	0xF	B4
debugErrorCode	Number	Unique error code corresponding to the crash	0xF	B5
TXPowerIndex	Number	TX Power index used by the device	0xF	B5
shutdownCause	String	Shutdown cause. Values can be USER-ACTION LOW-BATTERY DOWNLINK-REQUEST BLE_REQUEST BLE_CONNECTED Unknown		
payload	string	The uplink payload	All	All
debugCrashInfo	String	Crash information in ASCII	0xF	B6-B24
activityCounter	Number	Every motion detection, the activity counter is increased	0x7	B5=1 B6-9
deviceConfiguration	Map <Integer ,Integer>	A couple with parameter Id and its value	0x7	B5=2 B6-30
bssids	Map <String ,Integer>	A couple with a BSSID and RSSI (signal strength). Field populated for successful WIFI or BLE scan.	0x3	B4[3-0]=(2, 8) B6-B34
dataAge	Number	The age of the WIFI or BLE scan.	-	B4[3-0]=(2,8) B5

bleBeaconFailure	String	The cause of BLE beacon failure. Values can be: BLE_NOT_RESPONDING INTERNAL_ERROR SHARED_ANTENNA_NOT_AVAILABLE SCAN_ALREADY_ON_GOING NONE_BEACON_DETECTED HARDWARE_INCOMPATIBILITY UNKNOWN	0x3	B4[3-0]=8 B5

Examples

Position message

This example shows the decoding of an uplink position message. Remind that the *driverEngine* should created first.

The LoRa payload to be parsed is the following: "03480089100219ffa80433e7050083ae"

```
------------------------------ Snippet begin --------------------------
String payload = "03480089100219ffa80433e7050083ae";

JsonNode AbeewayUplinkPayloadString =
   driverEngine.uplink(
      "abeeway:abeewayDriver:1.7.0",
      JsonNodeFactory.instance.objectNode(),
      payload);

AbeewayUplinkPayload abeewayUplinkPayload =
   driverEngine
      .getDefaultMapper()
      .readValue(AbeewayUplinkPayloadString.toString(), AbeewayUplinkPayload.class);

System.out.println(abeewayUplinkPayload); //

------------------------------ Snippet End --------------------------
```

The result gives:

```
------------------------------ Snippet begin --------------------------
com.abeeway.codec.AbeewayDriverDecoderTest - decoded:
```

‘AbeewayUplinkPayload{latitude=43.6185088, longitude=7.0510336, EHPE=19, messageType=MESSAGE_POSITION, fixAge=16, mode=PERMANENT, batteryLevel=0.0, ackToken=1, fixType=GPS_FIX, periodicPosition=false, temperature=25.3, userAction=0, appState=1, moving=false, onDemand=false, payload=03480089100219ffa80433e7050083ae}’

---------- Snippet end ----------

4 Encoding a downlink

Overview

The input of the encoder is a string using the JSON format.
The output of the encoder is the LoRa payload coded in hexadecimal. This output is compatible with the Abeeway Trackers using the version v1.7.

Each message should contain the following fields:

- *DownMessageType*: Type of downlink messages (string format). Acceptable values are:
  - *ON_DEMAND_POSITION*: Request a position
  - *CHANGE_TRACKER_MODE*: Change the operation mode of a tracker.
  - START_SOS_MODE: Start the SOS mode.
  - STOP_SOS_MODE: Stop the SOS mode.
  - DEBUG_COMMAND: Send debug commands
  - SET_PARAMETER: Modify tracker's parameter(s).
  - GET_DEVICE_CONFIG: Read the tracker configuration.
- *AckToken*: Random integer ranging from 0 to 15 and used by the tracker to acknowledge the downlink message. Refer to the abeeway-reference-guide-fw1.7.pdf document for more details on setting/processing this field.

Notes:

- ➢ Each field name should be coded as a string.
- ➢ Extra fields should be provided according to the downlink message type.

The following sections detail each downlink message types and their required fields.

Request a position

This message requests a position to the tracker. The request does not require extra fields.

Example

The Json data content of the request is :

```
------------------------------ Snippet begin --------------------------
"{\"DownMessageType\" : \"ON_DEMAND_POSITION\", \"AckToken\" : 0}";
------------------------------ Snippet end ----------------------------
```

Operational mode change

This message change the tracker operational mode. The request requires an extra field called *ModeValue*, which can take the following string values:

- *STANDBY*: Set the tracker in standby.
- *MOTION*: Set the tracker in motion tracking.
- *PERMANENT*: Set the tracker in permanent tracking.
- *START_END*: Set the tracker in start/end mode
- *ACTIVITY*: Set the tracker in activity tracking mode.

- *OFF*: Set the tracker in off mode. Note that the master tracker will interpret this mode as *STANDBY* (since it cannot be woke up using the button).

Example

The Json data content of the request is :

```
------------------------------ Snippet begin --------------------------
"{\"DownMessageType\" : \"CHANGE_TRACKER_MODE\", \"ModeValue\" : \"MOTION\", \"AckToken\"
: 1}";
------------------------------ Snippet end --------------------------
```

SOS start

This message turn on the SOS mode of the tracker. The request does not require extra fields.

Example

The Json data content of the request is :

```
------------------------------ Snippet begin --------------------------
"{\"DownMessageType\" : \"START_SOS_MODE\", \"AckToken\" : 2}";
------------------------------ Snippet end --------------------------
```

SOS stop

This message turn off the SOS mode of the tracker. The request does not require extra fields.

Example

The Json data content of the request is :

```
------------------------------ Snippet begin --------------------------
"{\"DownMessageType\" : \"STOP_SOS_MODE\", \"AckToken\" : 3}";
------------------------------ Snippet end --------------------------
```

Debug command

This message performs debug actions on the tracker. The request requires an extra field called *DebugCommandType*, which can take the following string values:

- *RESET_DEVICE*: Reset the tracker.
- *READ_CURRENT_ERROR_AND_SEND_IT*: Read the last registered error and send it via LoRa.
- *TRIGGER_AN_ERROR*: Generate an error. Internal use only.

- *BLE_BOND_REMOVE*: Delete the BLE bonding.
- TRIGGER_HEARTBEAT_MESSAGE: Trigger a heartbeat message
- READ_TX_POWER_INDEX: Read TX Power Index and send a TX power debug uplink
- WRITE_TX_POWER_INDEX: Write New TX Power Index (For this type of debug command, a TxPowerIndex value is needed)

Example

The Json data content of the request :

---------- Snippet begin ----------

"{\"DownMessageType\" : \"DEBUG_COMMAND\", \"DebugCommandType\" : \"RESET_DEVICE\", \"AckToken\" : 2}";

---------- Snippet end ----------

Write New TX Power Index

This message performs write new TX Power Index . The request requires an extra field called *DebugCommandType* containing "*WRITE_TX_POWER_INDEX*" and a second the field called TxPowerIndex containing the new value of TX Power Index.

Example

The Json data content of the request is :

---------- Snippet begin ----------

"{\"DownMessageType\" : \"DEBUG_COMMAND\", \"DebugCommandType\" : \"WRITE_TX_POWER_INDEX\", \"AckToken\" : 2, \"TxPowerIndex\" :2}";

---------- Snippet end ---

Parameter modification

This message allows the modification of a configuration parameter. It requires systematically an extra field called *ParameterName* containing the string name of the parameter and a second field containing the new value. The following table provides the list of accepted parameter names as well as the second field name and its value.

Parameter name	Second field name	Second field value
GEOLOC_SENSOR_PROFILE	GeolocationType	Acceptable values are (string format): *WIFI_ONLY*: Use only WIFI scans. *GPS_ONLY:* Use only GPS. *XGPS_ONLY*: Use GPS and LP-GPS. *WIFI_FALLBACK_XGPS*: No more used. *SELF_GOVERNING_HISTORY*: No more used. *SELF_GOVERNING_TIMEOUT*: Superseded by WGPS_WXGPS_CHANGE_ON_TIMEOUT. *WGPS_ONLY*: WIFI then GPS if WIFI failed in one geolocation cycle.

		WXGPS_ONLY: WIFI then LP-GPS if WIFI failed in one geolocation cycle. *WGPS_WXGPS_CHANGE_ON_FAILURE*: No more used. *WGPS_WXGPS_CHANGE_ON_TIMEOUT*: Alternate WIFI/LP-GPS/GPS technologies. *BLE_BEACON_ONLY*:Only BLE beacon scans.
ONESHOT_GEOLOC_METHOD	GeolocationType	Acceptable values are (string format): *WIFI_ONLY*: Use only WIFI scans. *GPS_ONLY:* Use only GPS. *XGPS_ONLY*: Use GPS and LP-GPS. *WGPS_ONLY*: WIFI then GPS if WIFI failed in one geolocation cycle. *WXGPS_ONLY*: WIFI then LP-GPS if WIFI failed in one geolocation cycle. *BLE_BEACON*:Only BLE beacon scans.
EXT_ANTENNA_PROFILE	AntennaProfile	Acceptable values are (String format): *PRINTED_ANTENNA:* PCB antenna. *CERAMIC_ANTENNA:* Ceramic antenna. *DYNAMIC:* Dynamic antenna selection.
TRANSMIT_STRAT	TransmitStrat	Acceptable values are (string format): *SINGLE_FIXED*: Single TX. Use provisioned data rate. *SINGLE_RANDOM*: Single TX. Rate in [SF7..SF12]. *DOUBLE_FIXED*: First TX in [SF7..SF8], next use provisioned data rate. *DOUBLE_RANDOM*: First TX with rate in [SF7..SF8], next TX with rate in [SF9..SF12]. *NETWORK_ADR*: The LoRa network controls the data rate and the number of transmissions.
TRACKING_UL_PERIOD	ParameterValue	Integer value (number of seconds) in the range [60..86400] for EU and [30..86400] for US.
LORALIVE_PERIOD	ParameterValue	Integer value (number of seconds) in the range [300..86400].
ENERGY_STATUS_PERIOD	ParameterValue	Integer value (number of seconds) in the range [300..604800].
PERIODIC_POSITION_PERIO	ParameterValue	Integer value (number of seconds) in the range

D		[900..604800].
MOTION_START_END_NB_TX	ParameterValue	Integer value.
GPS_TIMEOUT	ParameterValue	Integer value (number of seconds) in the range [30..300].
XGPS_TIMEOUT	ParameterValue	Integer value (number of seconds) in the range [30..300].
GPS_EHPE	ParameterValue	Integer value (number of meters) in the range [0..100].
GPS_CONVERGENCE	ParameterValue	Integer value (number of seconds) in the range [0..300].
BLE_BEACON_COUNT	ParameterValue	Integer value in the range [1..4].
BLE_BEACON_TIMEOUT	ParameterValue	Integer values (number of seconds) in the range [1..5].
GPS_STANDBY_TIMEOUT	ParameterValue	Integer value (number of seconds) in the range [10..7200].
CONFIRMED_UL_BITMAP	ParameterValue	Integer value
CONFIRMED_UL_RETRY	ParameterValue	Integer value in the range [0..8].
CONFIG_FLAGS	FramePendingMechanism	Boolean value: true, false
CONFIG_FLAGS	ButtonPressToTurnOFF	Boolean value: true, false
CONFIG_FLAGS	SOSMode	Boolean value: true, false
CONFIG_FLAGS	DownlinkSetParamConfirmation	Boolean value: true, false
CONFIG_FLAGS	WifiPayloadWithNoCypher	Boolean value: true, false
CONFIG_FLAGS	BleAdvertising	Boolean value: true, false
CONFIG_FLAGS	GeolocStart	Boolean value: true, false
CONFIG_FLAGS	LedBlinkWhenGPSFix	Boolean value: true, false

Examples

1. Modify the geoloc sensor parameter:

---------- Snippet begin ----------

{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"GEOLOC_SENSOR_PROFILE\", \"GeolocationType\" : \"WIFI_ONLY\", \"AckToken\" : 10}";

---------- Snippet end ----------

2. Modify the geoloc method parameter:

---------- Snippet begin ----------

"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"ONESHOT_GEOLOC_METHOD\", \"GeolocationType\" : \"GPS\", \"AckToken\" : 8}";

---------- Snippet end ----------

3. Modify the antenna (industrial tracker having ceramic antenna):

---------- Snippet begin ----------

"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"EXT_ANTENNA_PROFILE\", \"AntennaProfile\" : \"CERAMIC_ANTENNA\", \"AckToken\" : 8}";

---------- Snippet end ----------

4. Modify the LoRa transmit strategy:

---------- Snippet begin ----------

"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"TRANSMIT_STRAT\", \"TransmitStrat\" : \"DOUBLE_FIXED\", \"AckToken\" : 1}";

---------- Snippet end ----------

5. Modify the position reporting period of the operational mode:

---------- Snippet begin ----------

"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"TRACKING_UL_PERIOD\", \"ParameterValue\" : 86400, \"AckToken\" : 10}";

---------- Snippet end ----------

6. Modify the LoRa heartbeat period:

---------- Snippet begin ----------

"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"LORALIVE_PERIOD\", \"ParameterValue\" : 3600, \"AckToken\" : 10}";

7. Modify the energy status reporting period:

---------- Snippet begin ----------

"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" :

\"ENERGY_STATUS_PERIOD\", \"ParameterValue\" : 604800, \"AckToken\" : 10}";

---------- Snippet end ----------

8. Modify the periodic position period frequency:

---------- Snippet begin ----------

"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" :\"PERIODIC_POSITION_PERIOD\", \"ParameterValue\" : 900, \"AckToken\" : 10}";

---------- Snippet end ----------

9. Configure the double press of the micro-tracker to trigger a SOS:

---------- Snippet begin ----------

"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"CONFIG_FLAGS\", \"SOSMode\" : true, \"AckToken\" : 5}";

---------- Snippet end ----------

10. Configure the double press of the micro-tracker to trigger an alert:

---------- Snippet begin ----------

"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"CONFIG_FLAGS\", \"SOSMode\" : false, \"AckToken\" : 5}";

---------- Snippet end ----------

11. Enable/Disable the off mode of the micro tracker (press the button to go in off mode):

---------- Snippet begin ----------

"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"CONFIG_FLAGS\", \"ButtonPressToTurnOFF\" : true,\"AckToken\" : 5}";

---------- Snippet end ----------

Note: Replace *true* by *false* to disable the off mode.

12. Enable/disable the frame pending mechanism:

---------- Snippet begin ----------

"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"CONFIG_FLAGS\", \"FramePendingMechanism\" : true,\"AckToken\" : 5}";

---------- Snippet end ----------

Note: Replace *true* by *false* to disable the frame pending mechanism.

13. Enable/Disable configuration confirmation uplink

---------- Snippet begin ----------

"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"CONFIG_FLAGS\", \"DownlinkSetParamConfirmation\" : true,\"AckToken\" : 5}";

---------- Snippet end ----------

Note: Replace *true* by *false* to disable the configuration confirmation uplink.

14. Enable/Disable the WIFI position message to be unciphered.

---------- Snippet begin ----------

```
"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"CONFIG_FLAGS\",
 \"WifiPayloadWithNoCypher\" : true, \"AckToken\" : 5}";
```

---------- Snippet end ----------

15. 0- Disable/Enable geoloc start event message when starting a geoloc sequence.

---------- Snippet begin ----------

```
"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"CONFIG_FLAGS\",
 \"GeolocStart\" : true, \"AckToken\" : 5}";
```

---------- Snippet end ----------

16. 0- Enable/disable LED Blink when a GPS fix is received event message when starting a geoloc sequence.

---------- Snippet begin ----------

```
"{\"DownMessageType\" : \"SET_PARAMETER\", \"ParameterName\" : \"CONFIG_FLAGS\",
 \"LedBlinkWhenGPSFix\" : true, \"AckToken\" : 5}";
```

---------- Snippet end ----------

Note: Replace *true* by *false* to disable the configuration confirmation uplink.

Parameter read

The codec provides a smart way to retrieve the tracker configuration. With a single command (downlink), you can either read the value of one or more parameters or retrieve the whole configuration.
The request (downlink message) is built as usual with the *DownMessageType* field set to the string value *GET_DEVICE_CONFIG*.
An optional field called *ListParameterID* can be also provided:

- When provided: its value contains the list of parameter identifiers that is expected.
- When omitted: the full configuration will be requested.

Note that, the tracker will answer this downlink message with one or more uplinks. Each related uplinks will have the *messageType* set to *MISC_DATA* and the *miscDataTag* set to *DEVICE_CONFIGURATION.* The field *deviceConfiguration* will be filled with the parameters list and their values.
The tracker is able to send up to 5 parameters value in one uplink. This means that if more than 5 parameters (or the full configuration) is requested, the tracker will send the required number of uplinks to fulfill the request.

The following table provide the mapping between the parameter names used by the decoder, the names used by the tracker and the identifiers.

Parameter identifier	Decoder name	Config name
Configuration parameters		
0	TRACKING_UL_PERIOD	ul_period
1	LORALIVE_PERIOD	lora_period
2	ENERGY_STATUS_PERIOD	pw_stat_period
3	PERIODIC_POSITION_PERIOD	periodic_pos_period
4	-	NA
5	GEOLOC_SENSOR_PROFILE	geoloc_sensor
6	ONESHOT_GEOLOC_METHOD	geoloc_method
7	EXT_ANTENNA_PROFILE	antenna
8	MOTION_START_END_NB_TX	motion_nb_ps
9	GPS_TIMEOUT	gps_timeout
10	XGPS_TIMEOUT	agps_timeout
11	GPS_EHPE	gps_ehpe

12	GPS_CONVERGENCE	gps_convergence
13	CONFIG_FLAGS	config_flags
14	TRANSMIT_STRAT	transmit_strat
15	BLE_BEACON_COUNT	BLE_beacon_count
16	BLE_BEACON_TIMEOUT	BLE_beacon_timeout
17	GPS_STANDBY_TIMEOUT	gps_standby_timeout
18	CONFIRMED_UL_BITMAP	confirmed_ul_bitmap
19	CONFIRMED_UL_RETRY	confirmed_ul_retry
Special parameters		
253	-	BLE firmware version
254	-	Application firmware version

Examples

1. Request the full configuration

---------- Snippet begin ----------

```
"{\"DownMessageType\" : \"GET_DEVICE_CONFIG\",\"AckToken\" : 1}";
```

---------- Snippet end ----------

2. Request only the parameters having the identifier 1,2,3,4 and 5.

---------- Snippet begin ----------

```
"{\"DownMessageType\" : \"GET_DEVICE_CONFIG\", \"ListParameterID\" : [1,2,3,4,5], \"AckToken\" : 2}";
```

---------- Snippet end ----------

Examples

Change mode message

This example shows the encoding of a downlink message for changing the mode to MOTION_TRACKING. Remind that the *driverEngine* should created first.

---------- Snippet begin ----------

```
String plainData = "{\"DownMessageType\" : \"CHANGE_TRACKER_MODE\", \"ModeValue\" : \"MOTION\", \"AckToken\" : 0}";
RawDownlink encodedPayload = driverEngine.downlink(
        "abeeway:abeewayDriver:1.7.0",
        JsonNodeFactory.instance.objectNode().put("plainDataFormat", true),
        driverEngine.getDefaultMapper().valueToTree(plainData));
System.out.println(encodedPayload);
```

---------- Snippet End ----------

The result gives:

---------- Snippet begin ----------
"020001"
---------- Snippet end ----------