TEKTELIC Communications Inc.
7657 10th Street NE Calgary, Alberta
Canada, T2E 8X2

TEKTELIC COMMUNICATIONS INC.

Document Type:	**Technical Reference Manual**
Document Number:	**T0006377_TRM**
Document Version:	**0.14**
Product Name:	**Custom Display Tablet**
Product Codes:	**T0006086 – Battery Powered Digital Signage** **T0006093 – Externally Powered Digital Signage**

TEKTELIC Communications Inc.
7657 10th Street NE
Calgary, AB, Canada T2E 8X2
Phone: (403) 338-6900

Revision History

Version	Date	Editor	Comments
0.1	May 23, 2018	Maulin Lodhia	First draft
0.2	May 28, 2018	Maulin Lodhia	Addressed review comments
0.3	May 30, 2019	Maulin Lodhia	Updated sections: 2.2, 3.1.3.2, 3.1.2.3 High level update: Now following downlink acknowledgement from the booking is followed by a get room status response message • Book now response • Extend meeting response • Book meeting in future response
0.4	July 19, 2019	Maulin Lodhia	Updated section 3.1.3.4 □ Added a byte for total capacity
0.5	Nov 15, 2019	C Karperien	Template Updates and adding Product Codes
0.6	Feb 5, 2020	C Karperien	Changed product name
0.7	April 27, 2020	Maulin Lodhia	Added section 3.1.3.2 MRDT custom hardware configuration
0.8	June 5, 2020	Maulin Lodhia	Updated section 3.1.3.2 with MRDT custom LED color mixing (RGB – Red, Green, and Blue) downlink message for Available and occupied room.
0.9	June 26, 2020	Maulin Lodhia	Cosmetic changes and changed default side LED color table
0.10	July 7, 2021	Tim Makarov	Added common customizable architecture
0.11	September 23, 2021	Tim Makarov	Updated common customizable architecture
0.12	September 28, 2021	Tim Makarov	Updated common customizable architecture and sections 3.1.3.1, 3.1.3.4
0.13	April 05, 2022	Tim Makarov	Updated common customizable architecture
0.14	December 02, 2022	Ade Adegboye	- Added information how to disable uplinks during EoD sleep (2.2.1) - Added information on how to limit uplink request retries (2.2.1) - Added information on how to configure accelerometer sensitivity (3.1.3.2)

			- Added information on how to update multiple elements with a single downlink (3.1.3.10) - Added information on how to modify EoD sleeping period (3.1.3.11)

Table of Contents

List of Tables

List of Figures

Acronyms and Glossary

ABP	activation by personalization
ADR	adaptive data rate
bps	bits per second
CRC	cyclic redundancy check
DL	downlink
EoD	*End of day*
EU	European Union as an RF region for LoRaWAN
Flash memory	Non-volatile memory located on the Home Sensor, which contains application and configuration settings.
FW	firmware
ID	identity / identifier
IoT	Internet of things
ISM	industrial, scientific, and medical
LoRa	a patented “long-range” IoT technology acquired by Semtech
LoRaMAC	LoRaWAN MAC
LoRaWAN	LoRa wide area network (a network protocol based on LoRa)
LoRaWAN Commissioning	the unique device identifiers and encryption keys used for LoRaWAN communication (see LoRaWAN Specification [1] for more details).
LSB	least significant bit
LTC	lithium thionyl chloride (the chemistry of LTC batteries)
MAC	medium access control
MCU	microcontroller unit
MRDT	Meeting Room Display Tablet
CDT	Meeting Room Display Tablet
ms	millisecond(s)
MSB	most significant bit
NA	North America as an RF region for LoRaWAN
NS	network server
OTA	over-the-air
OTAA	OTA activation
Reg	register
RF	radio frequency
RFU	reserved for future use
RO	read-only
RTU	remote terminal unit
R/W	read/write

Rx **...............................**	receiver
SW **..............................**	software
TRM **.............................**	technical reference manual
Tx **................................**	transmitter
UL **...............................**	uplink

1 Overview

This TRM describes the configuration options supported by the Meeting Room Display Tablet (MRDT). This document is intended for a technical audience, such as application developers and integration engineers, with an understanding of the NS and its command interfaces.

This TRM is only applicable to the MRDT Sensor modules listed in Table 1-1.

The MRDT Sensor is an all-purpose LoRaWAN IoT sensor run on four AA batteries and packed into a small casing. The MRDT features EPD screen, LCD controller, LED controller, accelerometer, and battery monitor (Battery Gauge). The battery lifetime of the MRDT Sensor is estimated to be 1 year.[1] Table 1-1 presents the currently available MRDT Sensor models.

Table 1-1: Kona MRDT Modules

Product Code & Revision	Description	RF Region
T0006086	MODULE, DIGITAL SIGNAGE, BATTERY POWERED	US 902-928 MHz (ISM band) & EU 863-870 MHz (ISM band)
T0006093	MODULE, DIGITAL SIGNAGE, EXTERNALLY POWERED	US 902-928 MHz (ISM band) & EU 863-870 MHz (ISM band)

Information streams currently supported by the MRDT are as follows:

- UL stream, i.e., from the MRDT to the Network Server (N.S.)
    - Readings obtained from on-board transducers (***sent on LoRaWAN port 10***)
    - Response to configuration and control commands from the NS (***sent on LoRaWAN port 100***)
- DL stream, i.e., data from the NS to the MRDT
    - Changing the state of the Sensor's (digital) outputs, i.e., open/close them (***sent on LoRaWAN port 10***)
    - Configuration and control commands used to change the Sensor's behavior (***sent on LoRaWAN port 100***)

[1] [1] This is for status updates (LoRa Tx/Rx) every 10 minutes at room temperature, with4x AA LiFeS2 battery having a total capacity of 7 Ah. Variations to this estimate can occur depending on the ambient temperature, use case, battery capacity, and battery self-discharge rate. Estimate is based on typical use case: 10-hour work day with LoRa update every 10 mins and 10 different meetings displayed on tablet per day.

The default configuration of the MRDT Sensor for reporting transducer readings includes the following:

- Report the battery voltage once every 1 (one) hour.
- For Class A mode operation, query room status once every 10 minutes.

The MRDT has a USB port used for upgrading its firmware and configuration files such as upgrading tablet images and fonts. Please consult the user guide document for more for more information on how to connect the tablet to the computer via the USB port to perform these upgrades/updates.

2 UL Payload Formats

The UL frame streams (from the Sensor to the NS) that are currently supported by the MRDT are as follows.

- The readings obtained from on-board transducers as explained in Section 2.1– sent on **port 10**
- Request booking application messages on **ports xx** as explained in Section 2.2.
- if configured in uplinks.yml file, ULs containing Custom messages on custom port numbers as explained in Section 2.3

2.1 Frame Payload to Report Transducers Data

Each data field from the MRDT is encoded in a frame format shown in Figure 2-1. A big-endian format (MSB first) is always followed.

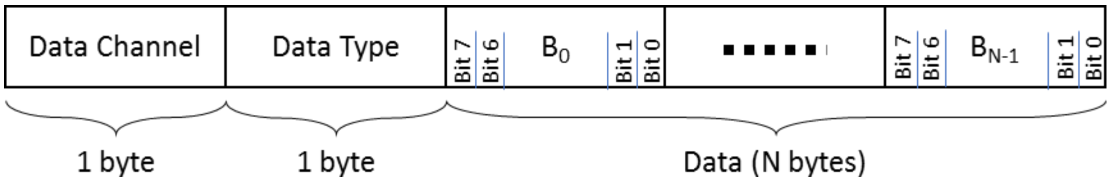


Figure 1: Frame Payload Block Format for ULs

A Sensor message payload can include multiple transducer data frames. These frames can also be arranged in any order. A single payload may include data from any given transducer. The MRDT frame payload values for transducers data are shown in Table 2-1. Transducers data in the UL are sent through ***LoRaWAN port 10***.

Table 2-1: UL Frame Payload Values for Transducers Data

Information Type	Data Channel	Data Type	Data Size [Bytes]	Data Type	Data Format
Battery Voltage	0x40	0xFF	2	Analog Input	Signed, 0.01 V/LSB
MCU Temperature	0x41	0x67	2	Temperature	0.1°C / LSB (signed)

2.1.1 Example Uplink Payloads

- 0x **40 FF** 01 2C **41 67** 00 0A
  - 0x **40 FF** (Battery Voltage) = (0x 01 2C) × 0.01 V = 3.00 V
  - 0x **41 67** (MCU Temperature) = (0x 00 0A) × 0.1°C = 1°C

2.2 Custom Messages

It is possible to add, remove or change existing configurations and customize user-specific uplinks to suit different applications. These changes are required to be made on uplinks.yml configuration file, and corresponding changes may be needed on the application layer to ensure the application is able to process the custom uplinks correctly. This process required to make changes to the uplinks are discussed below

2.2.1 Uplinks Configurations File

It is possible to add and change configurations for custom uplinks by modifying uplinks.yml configuration file if these changes are mirrored on the application to ensure smooth processing of the custom uplinks. The modified uplinks.yml file can be loaded on the tablet using the USB port on the MRDT. Please refer to the user guide document for instructions on how to do this.

Note: It is important to always have only one copy of the uplinks.yml configuration file in the tablet. Care must therefore be taken to ensure duplicate configuration files are therefore deleted before turning on the MRDT to ensure smooth operations of the device.

Uplink configuration file is based on YAML language and the structure of the uplinks.yml file is as seen below

```
---
Uplinks:
-
   <uplink 1 configurations>
-
   <uplink 2 configurations>
…
-
   <uplink N configurations>
```

Figure 2: Uplinks configuration file format

- The uplinks.yml document always starts with "---"which indicates the beginning of the configuration file. This is immediately followed by "Uplinks:" on the next line below
- Definition of a new uplink configuration block starts with a "-" on a separate line below the "Uplinks:" line
- **<uplink 1 configurations>**, **<uplink 2 configurations>** and **<uplink N configurations>** are field blocks of uplink 1, uplink 2, and up to uplink N configurations, where N < 255
- For each uplink configuration to be valid, the mandatory fields must be present and valid
- Optional fields as their group name suggests are not required to be filled for valid operation.

Note: Do not use symbols "#", "!" and "@" which are special symbols in YAML (for example "#" is a symbol of the beginning of commentary in YAML) for values! This may cause to unrecognizable configuration file or its part. You may use "!" and "@" symbols (but not "#" symbol) for string values inside double quotes.

Table 2-2: uplinks.yml fields

Field Type	Field Name	Field Information	Field Values and Limits
Mandatory	header_id	1 byte Message ID. Field value can be set to decimal or in hexadecimal (by using 0x or 0X prefix). 1 byte	[0x00 to 0x13, from 0x24 to 0x27, from 0x40 to 0x42, 0x70 and 0x72] are all ignored because they are predefined
	port	Defines the LoRa port number the UL will be sent on	MRDT ignores port value set to 20. All other values are accepted
	size	Defines the payload size (including the header_id)	If this is set to 0, MRDT sets it to 1 by default.
	source	Type of trigger for uplink. Possible types are: 1. ***timer:*** uplink will be sent periodically. If *source* is set as *timer*, *period* field becomes mandatory 2. ***element:*** uplink will be sent after pressing certain screen	MRDT ignores any other entries but the following: - timer - element - event

		button. If *source* is set as *element*, *element_id* becomes mandatory 3. ***event:*** uplink will be sent after a predefined event happens during operation. If *source* is set as *element*, *event_id* becomes mandatory	
Optional	period	Number of core ticks between two periodical uplinks. This field will be ignored if *source* is not set as *timer*.	MRDT ignores this field if source field is not set as timer
	element_id	Identifier for screen element or button that triggers an UL. All element_ids are stored in the elements.yml config file	This field will be ignored if source is not set as element. MRDT only accepts valid element_id as defined in 3.1.3.10
	event_id	Identifier for event that triggers UL. Acceptable values are: join_evt: triggers UL right after MRDT joins network additionalStatus_evt: all situations when room status request should be resent after booking operations	MRDT ignores this field if source field is not set as event. All values are ignored except the following - join_evt -additionalStatus_evt
	retry	Defines how often the periodic UL retries and their corresponding dummy packets should be resent when there is no response from NS	if retry = 0: MRDT does not retry if no response is received If retry = 1: MRDT retries indefinitely until response is received

			If 2 < retry < 254: MRDT retries for the number of times defined, then stops retrying even a response was not received Values outside the range 0 – 254 are ignored by the tablet.
	response_dl	Message ID of response downlink. The downlink with such message ID (any downlink with such message ID) stops uplink retry cycle and marks uplink request as responded It is allowed to set the value of this field in Decimal (should be set as unsigned integer number) or in Hexadecimal (should be set with 0x or 0X prefix)	This field will be ignored if *retry* field value is set equal to 0 or if there are no *retry* in uplink configurations.
	payload	1 byte payload value. It is only allowed to set the value of this field in Hexadecimal (should be set with 0x or 0X prefix)	This field is only valid and accepted if the value of the *size* field is greater than 2. It is possible to use different payloads for uplinks with the same header.
	disableInEodSleep	this is used to disable uplinks during end of day sleep. If set to 1, the UL will be disabled during End of day sleep to save battery life. If set to 0 or omitted, the uplink is enabled during end of day sleep.	MRDT ignores any other values apart from 0 and 1

```
---
Uplinks:

# Get Room Info request (0x38)
-
   header_id: 0x38
   port: 10
   size: 1 source:
   event
   event_id: join_evt
   retry: 1
   response_dl: 0x38
   disableInEodSleep: 1

# Get Room Status request (0x33)
-
   header_id: 51
   port: 10
   size: 1 source:
   timer period: 2
   retry: 1
   response_dl: 0x33
   disableInEodSleep: 0

# Book now request 15m (0x34)
-
   header_id: 0X34
   port: 10
   size: 2
   source: element
   element_id: btn_15mBookNow
   payload: 0x01
   retry: 1
   response_dl: 0x34
```

The example above shows an uplinks.yml configuration file for 3 uplinks (with message IDs *0x38*, *51* (that is 0x33) and *0x34* as set in their *header_id*).

All these uplinks start with a commentary line (string after # is interpreted as commentary in YAML). These commentaries give short optional briefs about the uplink that is configured in the block below.

All these uplinks are configured to be sent on port *10*.

Get Room Info request with *header_id 0x38* and Get Room Status request with *header_id 51* have *size 1* that means that these requests are consist of 1 byte message ID only. Book now request 15m with *header_id 0X34* has *size* equal to *2* that means that this uplink consists of 1 byte message ID and 1 byte payload. This payload is defined by *payload* (*0x01* in this case).

All these uplinks configured to be resendable indefinitely (*retry* field value is *1*) till the MRDT gets the mandatory response from NS (response downlink message ID is defined with *response_dl*).

Get Room Info request has *event* as its *source*. That means that Get Room Info request uplink is triggered by MRDT after special event in tablet. This event is defined in *event_id* field (*join_evt).* This means that this request is triggered by MRDT after successful joining the LoRa network.

Get Room Status request has *timer* as its *source*. That means that Get Room Status request uplink is triggered periodically by tablet every *2* core ticks that is set in *period*.

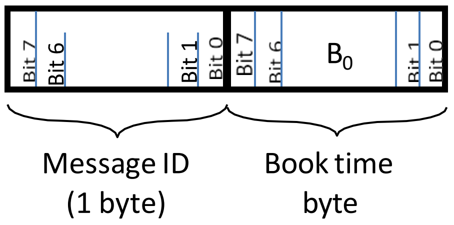
Book now request 15m has *element* as *source*. That means that Book now request 15m uplink is triggered by tablet after pressing the screen button that is set in *element_id* field (*btn_15mBookNow* in this case).

2.3 Frame Payload to Request Booking Application messages

Request of Tempo Application messages is one of the types of custom messages that can be configured on the MRDT. They will be triggered by MRDT if configured in uplinks.yml configuration file.

Each data field from the Sensor is encoded in a frame format shown in Figure 3 below. A big-endian format (MSB first) is always followed.

Figure 3:The UL frame payload format to request Booking App messages



The MRDT Sensor frame payload values for EPD soft button presses are shown in Figure 3.

Table 2-3: Request Booking Application Messages

Information Type	Data size without header (bytes)	Data Format
Room status request	0	
Book now request	1	Book meeting room for X minutes/hour Data value: 0x1 = 15 minutes 0x2 = 30 minutes 0x3 = 45 minutes 0x4 = 1 Hour
Extend meeting request	1	Book meeting room for X minutes/hour Data value: 0x1 = 15 minutes 0x2 = 30 minutes 0x3 = 45 minutes
Book meeting in future request	1	Book meeting room for X minutes/hour Data value: 0x1 = 30 minutes 0x2 = 45 minutes 0x3 = 1 Hour
Finish Meeting	0	
Room Information Request	0	
Room Amenities Status Request	0	

3 DL Payload Formats

The DL streams (from the Sensor to the NS) supported by the MRDT include,

- Configuration and control commands used to change the Sensor's behavior (sent on LoRaWAN **port 100)**
- Custom downlinks (sent on **custom ports** as defined in **dlinks.yml** file)

3.1 Configuration and Control Commands

A single DL configuration and control message can contain multiple command blocks, with a possible mix of read and write commands. Each message block is formatted as shown in Figure 3-1. A big-endian format (MSB first) is always followed.

The Command Field has a "register" address that is used to access various configuration parameters. These addresses are bound between 0x00 and 0x7F.

Bit 7 of the Command Field determines whether a read or write action is being performed. To write to a register, this bit must be set to 1 (one), but to read a register, it must be set to 0 (zero). All read commands are one-byte long. Data following a read access command will be interpreted as a new command block. Read commands are processed last. For example, in a single DL message, if there is a read command from a register and a write command to the same register, the write command is executed first.

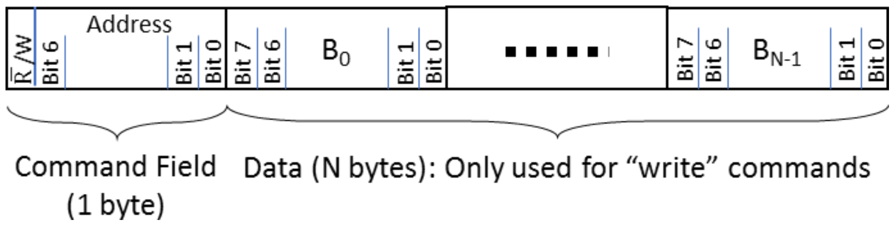


Figure 3-1: The format of a DL configuration and control message block.

All DL configuration and control commands are sent on ***LoRaWAN port 100***.

Examples:

In the following examples, the Command Field is boldfaced:

- Read Reg 0x00, 0x01, 0x02:
  - DL command: {0x **00 01 02}**
- Read Reg 0x05 and write value 0x8000 to Reg 0x10:
  - DL command: {0x **05 90** 80 00}

When a write command is sent to the MRDT, the MRDT immediately responds with a CRC32 of the entire DL payload as the first 4 bytes of the UL frame.

DL configuration and control commands fall into one of the following 4 (four) categories and are discussed in the following Sections.

- LoRaWAN Commissioning
- LoRaMAC Configuration
- Sensor Application Configuration
- Sensor Command and Control

3.1.1 LoRaWAN Commissioning

LoRaWAN commissioning values can be read back from the Sensor using DL commands. These registers are RO. See LoRaWAN 1.0.3 specification [1] for description of the values. Table 3-1 shows a list of these registers.

Table 3-1: LoRaWAN Commissioning Registers

Address	Access	Value	# Bytes
0x00	R	DevEUI	8
0x01	R	AppEUI	8
0x02	R	AppKey	16
0x03	R	DevAddr	4
0x04	R	NwkSKey	16
0x05	R	AppSKey	16

Note 1: Commissioning values need to be always kept secure.

Note 2: Registers 0x02, 0x04, 0x05 cannot be read back in some regions if the DR number is too small. For example, in the NA region, the maximum frame payload size with DR0 is 11 bytes.

3.1.2 LoRaMAC Configuration

LoRaMAC options can be configured using DL commands. These configuration options change the default MAC configuration that the Sensor loads on start-up. They can also change certain run-time parameters. Table 3-2 shows the MAC configuration registers. In this table, B_i refers to data byte indexed *i* as defined Figure 13

Table 3-2: LoRaMAC Configuration Registers

Address	Access	Value	# Bytes	Description
0x10	R/W	Join Mode	2	B_0-bit 7: 0 = ABP, 1 = OTAA B_1: RFU
0x11	R/W	• Unconfirmed/Confirmed UL • Disable/Enable Duty Cycle • Disable/Enable ADR	2	B0-bits 7–4: 0 = Class A, C = Class C B1-bit 0: 0 = Unconfirmed UL, 1 = Confirmed UL B1-bit 1 (RO): 0 = Private, 1 = Public Sync Word B1-bit 2: 0 = Disable duty cycle, 1 = Enable duty cycle B1-bit 3: 0 = Disable ADR, 1 = Enable ADR
0x12	R/W	• Default DR number • Default Tx Power number	2	B0-bits 3–0: Default DR number [2] B1-bits 3–0: Default Tx power number [2]
0x13	R/W	• Rx2 window DR number • Rx2 window channel frequency	5	B0-B1-B2-B3: Channel frequency in Hz for Rx2 B4: DR for Rx2
0x19	R/W	Net ID MSBs	2	Bytes B0-B1 in the Net ID (B0-B1-B2-B3)
0x1A	R/W	Net ID LSBs	2	Bytes B2-B3 in the Net ID (B0-B1-B2-B3)

Note: Modifying these values only changes them in the Sensor device. Options for the Sensor in the NS also need to be changed in order to not strand a Sensor. Modifying configuration parameters in the NS is outside the scope of this document.

Examples:

In the following example payloads, the Command Field is boldfaced:

- Switch Device to ABP Mode:
  - DL payload: {0x **90** 00 00}
- Set ADR On, No Duty Cycle, and Confirmed UL Payloads:
  - DL payload: {0x **91** 00 09}
- Set default DR number to 1 and default Tx Power number to 2:
  - DL payload: {0x **92** 01 02}

3.1.2.1 Default Configuration

Table 3-3 and Table 3-5 lists the default values for the LoRaMAC configuration registers (cf. [1], [2]).

Table 3-3:Default Values of LoRaMAC Configuration Registers

Address	Default Value
0x10	0x 80 00 (OTAA mode)
0x11	0x 00 0E (Class A, Unconfirmed UL, enabled duty cycle, enabled ADR)
0x12	0x 00 04 (DR0, Tx Power 0—max power, see Table 3-4)
0x13	As per Table 3-5.
0x19	0x 00 00
0x1A	0x 00 00

Table 3-4: Maximum Tx Power in Different Regions by Default

RF Region	Max Tx EIRP [dBm]
EU868	16
US915	30
AS923	16
AU915	30
IN865	30
CN470	19.15
KR920	14
RU864	16

Table 3-5: Default Values of Rx2 Channel Frequency and DR Number in Different Regions

RF Region	Default Value	Channel Frequency	DR Number
EU868	0x 33 D3 E6 08 00	869.525 MHz	DR0
NA915	0x 37 08 70 A0 08	923.3 MHz	DR8
AS923	0x 37 06 EA 00 02	923.2 MHz	DR2
AU915	0x 37 08 70 A0 08	923.3 MHz	DR8
IN865	0x 33 A6 80 F0 02	866.55 MHz	DR2
CN470	0x 1E 1E 44 20 00	505.3 MHz	DR0
KR920	0x 36 F3 13 E0 00	921.9 MHz	DR0
RU864	0x 33 CD 69 E0 00	869.1 MHz	DR0

3.1.3 MRDT Sensor Application Configuration

This section lists all possible Sensor application configurations (as part of DL configuration and control commands), like periodic Tx configuration, accelerometer sensitivity, and LED settings.

Note: Care must be taken to avoid stranding the sensor during reconfiguration. If all sensing inputs are disabled, the device will not be able to be reconfigured.

3.1.3.1 Periodic Tx Configuration

All periodic sensor reporting is synchronized around “ticks”. A tick is simply a user configurable time-base that is used to schedule sensor measurements. For each transducer, the number of elapsed ticks before transmitting can be defined, as shown in Table 3-6.

Table 3-6: Ticks Configuration for Periodic Tx

Address	Access	Value	# Bytes	Description
0x20	R/W	Seconds in a Tick	4	Sets the core tick in seconds for periodic events (0 disables)
0x21	R/W	Ticks per Battery Tx	2	Ticks between battery reports (0 disables)
0x22	R/W	Ticks per MCU Temperature Tx	2	Ticks between MCU temperature reports (0 disables)

3.1.3.1.1 Seconds in a Tick

All periodic transmit events are schedule in “ticks”. This allows for sensor reads to be synchronized, reducing the total number of uplinks required to transmit sensor data. The minimum number of seconds in a tick is 30 seconds and the maximum is 86400 seconds (i.e., a day). Values from 1 to 29 or above 86400 are RFU and will be ignored by the sensor. If “Seconds in a Tick” is set to 0 (zero), ***all*** periodic reporting will be disabled regardless of individual sensor reporting configurations. Disabling all periodic based reporting is not recommended!

3.1.3.1.2 Ticks per <Transducer>

Sets the individual tick period for a transducer. Once the configured number of ticks has expired the Sensor will poll the specified transducer and report the data in an uplink message.

A setting of 0 (zero) will disable periodic reporting for the specified transducer. Disabling all periodic based reporting is not recommended!

3.1.3.1.3 Default Configuration

Seconds in a Tick	300 seconds (5 min)
Ticks per Battery Tx	12 (1 hour)
Ticks per MCU Temperature Tx	0 (disabled)

3.1.3.1.4 Example DL Messages

- Disable all periodic events:
  - 0x: A0 00 00 00 00 (Reg 20, write bit set to true) – Seconds in a Tick = 0 (disabled)
- Read the current “Seconds in a Tick” value:
  - 0x: 20 (Reg 20, write bit set to false)
- Write “Tick per MCU Temperature Tx”:
  - 0x: A2 00 01 (Reg 22, write bit set to true) – set “Ticks per MCU Temperature Tx” to 1 (one).

3.1.3.2 MRDT custom hardware configuration

MRDT is equipped with side LEDs and Front EPD screen light. The LED color can be custom configured based on available or occupied room status. The front EPD screen light intensity level can also be custom configured. Please note that these settings are only available in externally powered versions. Side LED and front EPD screen light are disabled/switched OFF in battery-powered variants to save battery.

Table 3-7: MRDT LED and Screen Light hardware custom configuration

Address	Access	Value	# Bytes	Description
0x24	R/W	LED color setting for available room status	3	Available room status LED color setting (Red =0x00, Green = 0x00, Blue = 0x00 turns off LEDs) Each color/byte value ranges from 0x00 to 0xFF
0x25	R/W	LED color setting occupied room status	3	Occupied room status LED color (Red =0x00, Green = 0x00, Blue = 0x00 turns off LEDs) Each color/byte value ranges from 0x00 to 0xFF
0x26	R/W	Front light led intensity level	1	Front EPD screen light intensity level (0 disables, 100 Max Intensity) Valid Range: 0 – 100

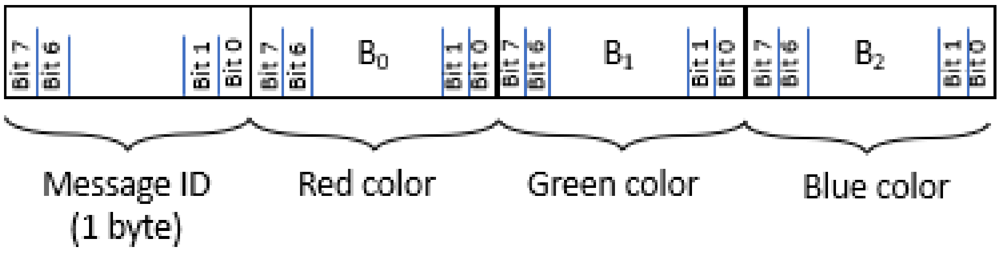
0x28	R/W	Accelerometer Sensitivity	2	Accelerometer Click threshold for waking up tablet from sleep Valid Range: 0 - 2032milli-g

3.1.3.2.1 Available/Occupied room status LED color

In externally powered MRDT the side LEDs are multi-colored and can be customized using red, green blue color combinations. In MRDT firmware users can customize LED color per room status (available or occupied).

The following the DL frame format for Available/Occupied room status LED color setting message.

Figure 4: The format of Available/Occupied room status LED color setting



3.1.3.2.2 Front EPD screen light intensity level

Sets the front EPD screen light intensity level.

A setting of 100 will set the front EPD screen light intensity level to maximum level. A setting of 0 (zero) will disable front EPD screen light.

3.1.3.2.3 Default Configuration

Available Room Status LED color	0x00 0x00 0x00 (OFF)
Occupied Room Status LED color	0xFF 0xA6 0x2C (White)
Front EPD screen intensity level	50

3.1.3.2.4 Example DL Messages

- Disable LED color:

    - 0x: A4 00 00 00 (Reg 24, write bit set to true) – Available room status LED color = 0x00 0x00 0x00 (disabled)
    - 0x: A5 00 00 00 (Reg 25, write bit set to true) – Occupied room status LED color = 0x00 0x00 0x00 (disabled)
- Read the current “Front EPD screen intensity level” value:
    - 0x: 26 (Reg 26, write bit set to false)
- Write “Available room status LED color” and “Occupied room status LED color”:
    - 0x: A4 FF A6 2C A5 00 00 FF (Reg 24 and Reg 25, write bit set to true) – set “Available room status LED color” to 0xFF 0xA6 0x2C (White) and “Occupied room status LED color” to 0x00 0x00 0xFF (Blue)

3.1.3.2.5 Accelerometer Sensitivity

This register sets the Accelerometer sensitivity threshold level for detecting wake-up screen tap. When set to the maximum value – 2032 milli-g, the touch screen will require a strong tap to wake up. Also. When set to the minimum value – 0 milli-g, the screen becomes highly sensitive to touch and vibrations.

3.1.3.2.6 Default Configuration

Accelerometer Sensitivity Threshold	**150 milli-g**

Example DL Messages

- Read the current “Accelerometer Sensitivity Threshold” value:
    - 0x: 28 (Reg 28, write bit set to false)
- Set “Accelerometer Sensitivity Threshold” to 50 milli-g:
    - 0x: A8 00 32
- Set “Accelerometer Sensitivity Threshold” to 1500 milli-g:
    - 0x: A8 05 DC

3.1.3.3 Custom downlinks

It is possible to add and change configurations for custom downlinks that are correspond to application in dlinks.yml configuration file. This file should be uploaded to MRDT through USB.

3.1.3.4 Downlinks configuration file

It is possible to add and change configurations for custom downlinks that are correspond to application in dlinks.yml configuration file. This file should be uploaded to the Tablet through USB. Downlink configuration file is based on YAML language. The structure of this file is:

Figure 5: Structure of dlinks.yml file

```
---
Downlinks:
-
  <downlink 1 configurations>
-
  <downlink 2 configurations>
…
-
  <downlink N configurations>
```

- The dlinks.yml document always starts with "---"which indicates the beginning of the configuration file. This is immediately followed by "Downlinks:" on the next line below
- Definition of a new downlink configuration block starts with a "-" on a separate line below the "Downlinks:" line
- **<downlink 1 configurations>**, **<downlink 2 configurations>** and **<downlink N configurations>** are field blocks of downlink 1, downlink 2, and up to downlink N configurations, where N < 255
- For each downlink configuration to be valid, the mandatory fields must be present and valid
- Optional fields as their group name suggests are not required to be filled for valid operation.

Note: Do not use symbols "#", "!" and "@" which are special symbols in YAML (for example "#" is a symbol of the beginning of commentary in YAML) for values! This may cause to unrecognizable configuration file or its part. You may use "!" and "@" symbols (but not "#" symbol) for string values inside double quotes.

It is allowed to configure different downlinks with the same *header_id* or any other field. Mandatory fields in downlink configurations are:

- *header_id* – 1 byte message ID. It is allowed to set the value of this field in Decimal (should be set as usual number) and in Heximal (should be set with 0x or 0X prefix).

- *port* – LoRa port number.

  Optional fields in downlinks configurations are:

- *handler_id* – identifier of the handler that processes downlink payload. Possible values are:
  - *roomInfo_hdl* – handler for Get room information response downlink (see 3.1.3.8).
  - *roomStatus_hdl* – handler for Get room status response downlink (see 3.1.3.7).
  - *bookNow_hdl* – handler for Book now response downlink (see 3.1.3.6).
  - *bookNext_hdl* – handler for Book meeting in future response downlink (see 3.1.3.6).
  - *extend_hdl* – handler for Extend meeting response downlink (see 3.1.3.6).
  - *finish_hdl* – handler for Finish meeting response downlink (see 3.1.3.6).
  - *roomAmnities_hdl* – handler for Get amenities status response downlink (see 3.1.3.9).
  - *updateElement_hdl* – handler for single update element downlink (see 3.1.3.10).
  - *updateSeveralElements_hdl* – handler for multiple update element downlink (see 3.1.3.10).
  - *eodSleep_hdl* – handler for End-of-day sleep downlink (see 3.1.3.11).

- *screen_id* – identifier of the screen that tablet should switch to after receiving configured downlink. Possible values are:
  - *init_scr* – initialization screen that consist of 6 labels, 1 panel and 1 image. This screen is designed to view base Meeting Room Display Tablet version information.
  - *avail_scr* – screen that is designed to view that room is available during Meeting Room Display Tablet operations in connection with Booking application. It consists of 8 labels, 1 button, 1 panel and 6 images.
  - *availBookNow_scr* – screen that is designed to give possibility to book the room from screen of the Meeting Room Display Tablet during tablet operations in connection with Booking application. It consists of 6 labels, 4 buttons, 1 panel and 6 images.
  - *occup_scr* – screen that is designed to view that room is occupied during Meeting Room Display Tablet operations in connection with Booking application. It consists of 10 labels, 3 buttons, 1 panel and 6 images.
  - *occupExtend_scr* – screen that is designed to give possibility to extend the meeting from the screen of the tablet during Meeting Room Display Tablet operations in connection with Booking application. It consists of 8 labels, 4 buttons, 1 panel and 6 images.
  - *occupFinish_scr* – screen that is designed to give possibility to finish the meeting from the screen of the tablet during Meeting Room Display Tablet operations in connection with Booking application. It consists of 8 labels, 2 buttons, 1 panel and 6 images.
  - *occupBookNext_scr* – screen that is designed to give possibility to book meeting in future from the screen of the tablet during Meeting Room Display Tablet operations in connection with Booking application. It consists of 8 labels, 4 buttons, 1 panel and 6 images.
  - *occupBookNowSuccess_scr* – screen that is designed to view if Book now request was

    acked during Meeting Room Display Tablet operations in connection with Booking application. It consists of 6 labels, 1 panel and 6 images.
    - *occupBookNextSuccess_scr* – screen that is designed to view if Book meeting in future request was acked during Meeting Room Display Tablet operations in connection with Booking application. It consists of 8 labels, 1 panel and 6 images.
    - *occupExtendSuccess_scr* – screen that is designed to view if Extend meeting request was acked during Meeting Room Display Tablet operations in connection with Booking application. It consists of 8 labels, 1 panel and 6 images.
    - *custom1_scr* – first screen that is designed to have possibility to configure the custom screen without connection to Booking application. It consists of 10 labels, 6 buttons, 5 panels and 5 images. Only custom screens do not have cross- screen connections between labels, panels, and images.
    - *custom2_scr* – second screen that is designed to have possibility to configure the custom screen without connection to Booking application. It consists of 10 labels, 6 buttons, 5 panels and 5 images. Only custom screens do not have cross- screen connections between labels, panels, and images.
    - *custom3_scr* – third screen that is designed to have possibility to configure the custom screen without connection to Booking application. It consists of 10 labels, 6 buttons, 5 panels and 5 images. Only custom screens do not have cross- screen connections between labels, panels, and images.

      custom4_scr – forth screen that is designed to have possibility to configure the custom screen without connection to Booking application. It consists of 10 labels, 6 buttons, 5 panels and 5 images. Only custom screens do not have cross- screen connections between labels, panels, and images.

      NOTE: Only custom screens do not have cross-screen connections between labels, panels, and images.

- *ack_screen_id* – identifier of the screen that tablet should switch to after receiving acked configured downlink (see 3.1.3.6). This field is ignored if *screen_id* is present in downlink configurations. This field is used as *screen_id* configuration if *nack_screen_id* and *screen_id* fields are absent in downlink configurations. Possible values are the same as for *screen_id*.

- *nack_screen_id* – identifier of the screen that tablet should switch to after receiving nacked configured downlink (see 3.1.3.6). This field is ignored if *screen_id* is present in downlink configurations. This field is used as *screen_id* configuration if *ack_screen_id* and *screen_id* fields are absent in downlink configurations. Possible values are the same as for *screen_id*.

NOTE: The number of configured downlinks in dlinks.yml file should not exceed 254.

IMPORTANT: Do not use symbols “#”, “!” and “@” which are special symbols in YAML (for example “#” is a symbol of the beginning of commentary in YAML) for values! This may cause to unrecognizable configuration file or its part. You may use "!" and "@" symbols (but not "#" symbol) for string values inside double quotes.

3.1.3.5 Example downlinks configuration file

Downlink configuration file example is:

Table 3-7: Example of dlinks.yml configuration file

```
---
Downlinks:

# Get Room Info response (0x38)
-
   header_id: 0x38
   port: 103
   handler_id: roomInfo_hdl
   screen_id: init_scr

# Get Room Status response (0x33)
-
   header_id: 51
   port: 102
   handler_id: roomStatus_hdl
   ack_screen_id: occup_scr
   nack_screen_id: avail_scr

# Room amenities status response (0x39)
-
   header_id: 57
   port: 104
   handler_id: roomAmnities_hdl
```

This example file includes configurations for 3 downlinks (with message IDs 0x38, 51 (that is 0x33) and 57 (that is 0x39) that are set in *header_id*).

All these downlinks are headed with commentary started with # sign. These commentaries give short brief about the downlink that is configured below, and they are optional.

Get Room Info response downlink (with *header_id* 0x38) is the first downlink in this configuration file. It is configured to port 103. This mean that optional configurations (*handler_id* and *screen_id* in this case) is used only if tablet received downlink on port that is set in *port* field and with message ID that is specified in field *header_id* (*port* 103 and *header_id*

0x38 in this case). The configurations for this downlink include *roomInfo_hdl* as value for *handler_id* field. This means that payload of this downlink is processes as Get Room Information response (see detailed about Get Room Information response in 3.1.3.8). There is *screen_id* with *init_scr* as its value in configurations of this downlink. This mean that tablet switches its view to initialization screen after receiving of this downlink.

Get Room Status response is configured as downlink with message ID 51 (or 0x33 in Heximal) that is set in *header_id* field and is waiting on port 102 that is set in *port* field. The payload of this downlink is processes as Get Room Status response (see detailed about Get Room Status response in 3.1.3.7) that is set in *handler_id* by value *roomStatus_hdl*. The view of the tablet switches to the screen that is designed to show that the room is occupied after receiving this acked downlink (that is set in field *ack_screen_id* by value *occup_scr*) and to screen that is designed to show that the room is available after receiving this nacked downlink (that is set in field *nack_screen_id* by value *avail_scr*) (see detailed about acked and nacked downlinks in 3.1.3.6).

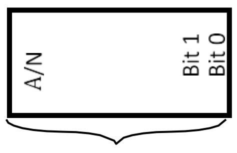
Get Room Amenities status response is configured as downlink with message ID 57 (or 0x39 in Heximal) that is set in *header_id* field and is waiting on port 104 that is set in *port* field. The payload of this downlink is processes as Get Room Amenities status response (see detailed about Get Room Amenities status response in 3.1.3.9) that is set in *handler_id* by value *roomAmnities_hdl*. The view of the tablet does not switch to any screen because there no *screen_id*, *ack_screen_id* or *nack_screen_id* in configurations for this downlink.

3.1.3.6 Booking application acknowledgements

The message ID field is the response for the message UL request (as described in Table 2-2).

The message block is formatted as shown in Figure 3-13. A big-endian format (MSB first) is always followed.

NOTE: Each acknowledgement is followed by a room status response as described later section 3.1.3.7



Message ID
(1 byte)

Figure 6: The format of a DL booking application acknowledgements

The following table describes the booking application downlink acknowledgments.

Table 3-8: DL Frame Booking Application acknowledgements

Information Type	**Example Message ID Field Value**	**Maximum DL Frame size (Bytes)**	**Number of DL messages per UL message request**	**Information**
Book now response (example *header_id* is 0x34)	0xB4	1	1	ACK
	0x34	1	1	NACK
Extend meeting response (example *header_id* is 0x35)	0xB5	1	1	ACK
	0x35	1	1	NACK
Book meeting in future response (example *header_id* is 0x36)	0xB6	1	1	ACK
	0x36	1	1	NACK
Finish meeting (example *header_id* is 0x37)	0xB7	1	1	ACK
	0x37	1	1	NACK

3.1.3.7 Booking application get room status response

A single UL get room status request may result in multiple DL responses and acknowledgements. Each message block is formatted as shown in Figure 3-13. A big-endian format (MSB first) is always followed.

The message ID field is the response for the message UL request (as described in Table 2-2).

Message ID bit 7 (A/N) of the message ID determines whether message is ack or nacked. For room status response, this bit determines whether room is occupied (bit value 1) or available (bit value 0).

Byte 0 (B_0), bit 7 (EPD_E) describes if EPD screen needs to be turned off. This is required for turning off the screen during off hours to save battery.

- ON (bit value of 1)
- OFF (bit value or 0)

Byte 0 (B_0), bit 6 (TS_E) describes if touch screen needs to be turned off. This is required for turning off the screen during off hours to save battery or locking the screen.

- ON (bit value of 1)
- OFF (bit value or 0)

NOTE: For battery powered units, when EPD_E and TS_E bits are OFF, the unit will go to deep sleep mode for 14-hour predefined timer by default. When same bit is turned on unit comes out of deep sleep. To come out of deep sleep user may tap the EPD screen to wake up but would go back to sleep for predefined timer of 30 seconds. The application may send room status response with EPD_E and TS_E bits turned ON before the 14-hour timer expiry to wake up from deep sleep, at this point the unit will be out of deep sleep mode and ready for regular operation. To summarize, the unit will stay in deep sleep mode for 14 hours. If user taps the screen in middle of this 14-hour period, unit will wake up for 30s and go back to deep sleep.
The application can wake up the unit before the 14-hour period by sending EPD_E and TS_E bits ON in room status response or by sending special End-of-day sleep downlink. The predefined timer of 14 hours can be configured to a desirable value (see section 3.1.3.11)

Byte 0 (B_0), bit 5 (Nx/C) describes what this DL frame contains

- Next booking information (bit value 1)

  Next booking information (Nx):

  - Next booking time (Byte B_0, B_1) HH:MM AM/PM
  - Booked by string size maximum of 30 bytes.
  - Booked by string - N bytes of string

- Current booking information (bit value 0).

  Current booking information (C):

  - Current booking till time (Byte B_0, B_1): HH:MM AM/PM
  - Booked by string size maximum of 30 bytes.
  - Booked by string - N bytes of string

Byte B_0, bit 4 determines the time is PM (bit value of 1) or AM (bit value 0)

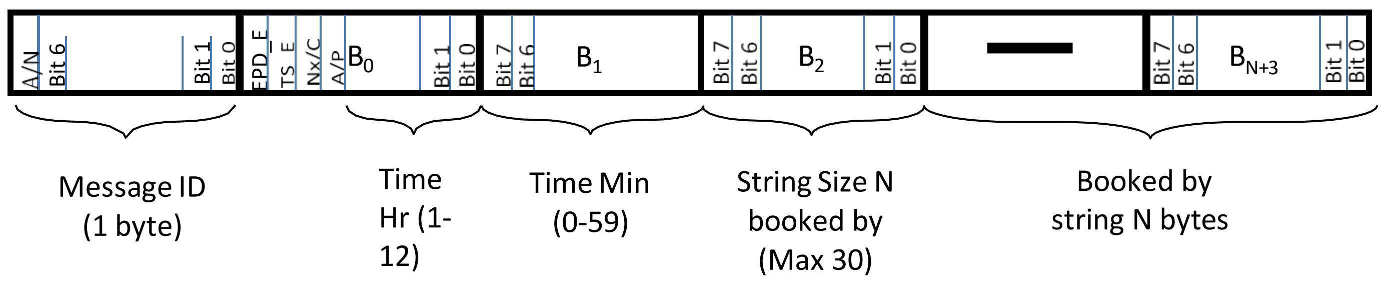


Figure 7: The format of a DL booking application 'get room status' message block.

The uplink booking application get room requests require response split into 2 DL frames.

First DL frame containing information of current meeting (till time, booked by) and following DL frame containing information about next meeting (next meeting book time, next meeting booked by).

It is necessary to send "Booked by string" as ASCII code of string "No Meeting" to view NO NEXT MEETING line on tablet screen instead of "by" line for future meeting.

It is necessary to send Byte B_2 value 0 to hide "by" line on tablet screen for current or for next booking and to not send "Booked by string".

The following table describes the booking application get room status downlink response message.

Table 3-9: DL Frame Booking Application get room status response.

Information Type	**Example Message ID Field Value**	**Maximum DL Frame size (Bytes)**	**Number of DL messages per UL message request**	**Information**
Room status response	0xB3	34	2	Occupied
(Example *header_id* is 0x33)	0x33	34	1	Available

3.1.3.8 Booking application get room information response/acknowledgement

This message response/acknowledgement is different from regular DL booking application acknowledgments described in earlier section.

The following the DL frame format for get room information response message.

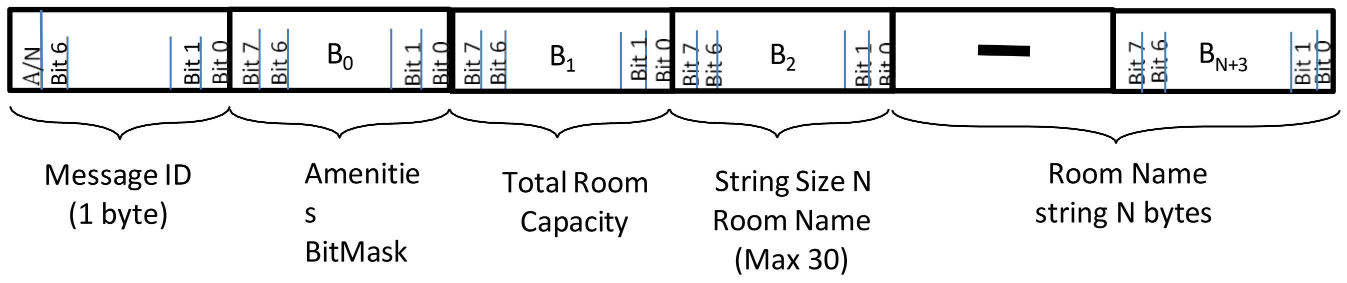


Figure 8: The format of a DL booking application get room information acknowledgment message block.

Message ID bit 7 (A/N) of the message ID determines whether message is ack or nacked. Tablet ignore this bit and use B_0 – B_{N+3} from ack or nacked message.

Byte 0 (B_0) for this response message contains the Amenities presence bit mask

- Bit 0 – TV
- Bit 1 – Projector
- Bit 2 – Web Camera
- Bit 3 – White Board

Byte 1 (B_1) is size of the Room name string N bytes.

Subsequent bytes are room name string of N bytes.

The following table describes the booking application get room information downlink acknowledgment.

Table 3-10: DL Frame Booking Application get room information acknowledgement.

Information Type	**Example Message ID Field Value**	**Maximum DL Frame size (Bytes)**	**Number of DL messages per UL message request**	**Information**
Room information response (example *header_id* is 0x38)	0xB8	34	1	ACK
	0x38	34	1	NACK

3.1.3.9 Booking application get amenities status response

This message response/acknowledgement is different from regular DL booking application acknowledgments described in earlier section.

The following the DL frame format for get room information response message.

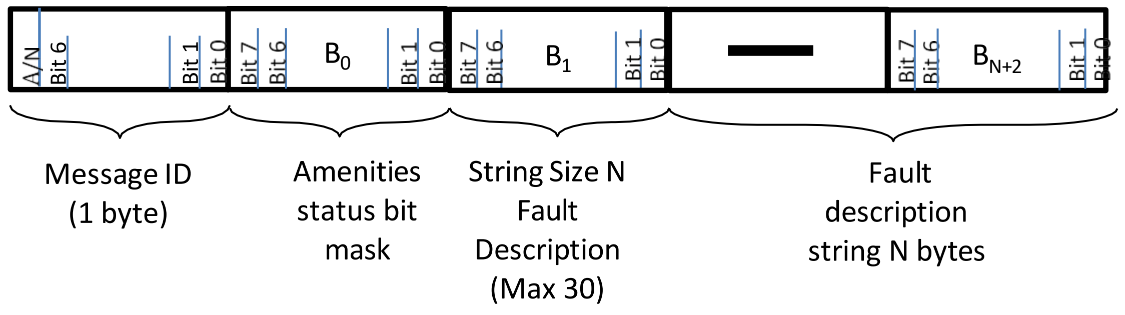


Figure 9: The format of a DL booking application get amenities acknowledgment message block

Message ID bit 7 (A/N) of the message ID determines whether message is ack or nacked.

Byte 0 (B_0) for this response message contains the Amenities status bit mask

- Bit 0 – TV
- Bit 1 – Projector
- Bit 2 – Web Camera
- Bit 3 – White Board

Bit value of 1 means the amenity is at fault.

Byte 1 (B_1) is size of the fault description string N bytes.

Subsequent bytes are fault description string of N bytes.

The following table describes the booking application get amenities status downlink acknowledgment.

Table 3-11: DL Frame Booking Application get amenities status acknowledgement

Information Type	**Example Message ID Field Value**	**Maximum DL Frame size (Bytes)**	**Number of DL messages per UL message request**	**Information**
Get Amenities status response (example *header_id* is 0x39)	0xB9	33	1	ACK
	0x39	1	1	NACK

3.1.3.10 Update screen element downlink

The tablet's screen elements can be updated by using one of the three formats below.

1. Format 1: Update a single screen element with text update

   This format is to be used only when a single screen element is to be updated. For example, updating the text value for a temperature value requires only changing the screen element for the temperature value using its unique screen ID. This format must be used with handler_id: updateElement_hdl definition in dlinks.yml file

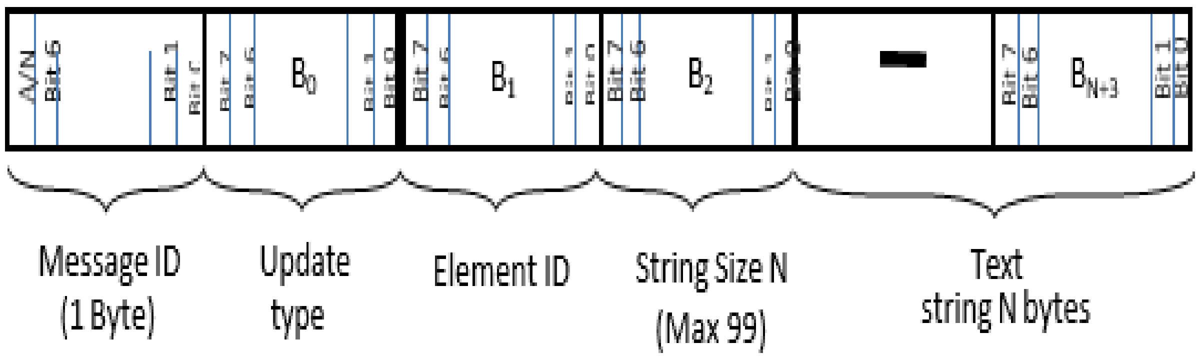
2. Format 2: Update a single screen element with visibility update

   Like Format 1 above, this format is only used to update the visibility of a screen element. For example, this format is used to hide a single screen element using its unique screen ID. This format must be used with handler_id: updateElement_hdl definition in dlinks.yml file

3. Format 3: Update multiple screen elements with text and/or visibility update
   This format is used for making changes to multiple screen elements with the same DL. It can be used to update texts and make elements visible/invisible using a single DL. This format must be used with handler_id: updateSeveralElements_hdl definition in dlinks.yml file

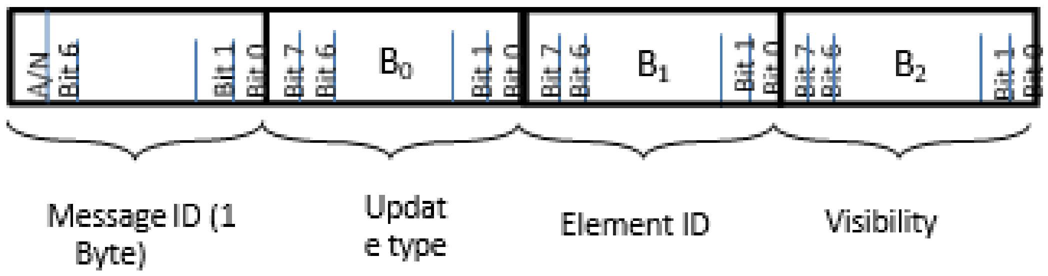
<u>Format 1</u>

Figure 10: DL format for updating single screen element with text update



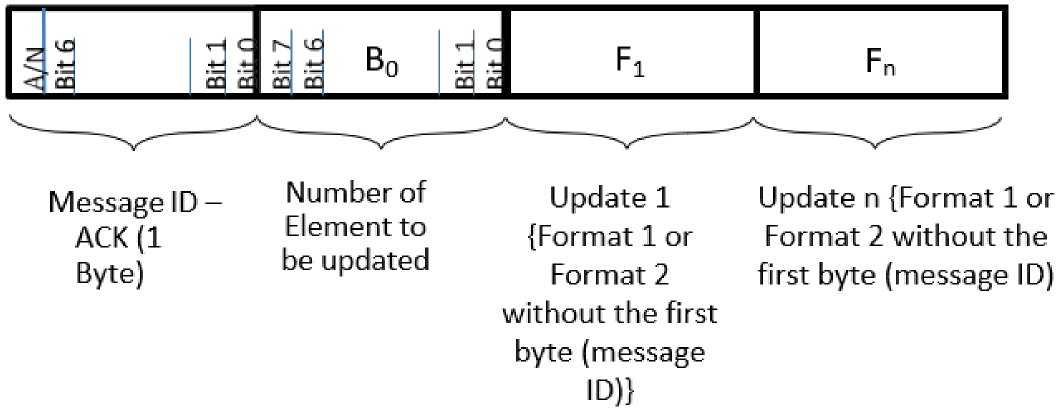
Format 2

Figure 11: DL format for updating single screen element with visibility update



Format 3

Figure 12: DL format for updating multiple screen elements with visibility and/or Text update


A/N
Bit 6
Bit 1
Bit 0
Bit 7
Bit 6
B_0
Bit 1
Bit 0
F_1
F_n
Message ID – ACK (1 Byte)
Number of Element to be updated
Update 1 {Format 1 or Format 2 without the first byte (message ID)}
Update n {Format 1 or Format 2 without the first byte (message ID)

Message ID bit 7 (A/N) of the message ID determines whether message is ack or nacked. Nacked message is ignored.

Format 1 and Format 2: B_0 is used to determine screen element update type. Possible values for B_0 are: 1 – label text update, 2 - screen element visibility update (message with any other value in B_0 is ignoring by tablet).

Format 3: B_0 is used to determine the number of screen elements to be updated. The maximum number of elements that can be updated with a single downlink depends on the network DR set. This means that the DL will accept as many updates as the set DR allows.

B_1 is determine screen element ID number (identifier of the element of the screen that necessary to update). Possible values for B_1 are corresponding with *id* field in elements.yml screen element configuration file is tabulated below. 0 is for *lab_productName*, 1 – *lab_fwVer*, 2 – *lab_bootloadVer*, 3 – *lab_apploadVer*, 4 – *lab_iteFwVer*, 5 – *lab_tsFwVersion*, 6 – *lab_loraStatus*, 7 – *img_logoBlack*, 8 – *pnl_bottom*, 9 – *lab_tapToWakeUp*, 10 – *lab_bookPending1*, 11 – *lab_nextBook1*, 12 – *lab_nextBook2*, 13 – *lab_bookBy*, 14 – *lab_roomName*, 15 – *lab_available*, 16 – *lab_persons*, 17 – *btn_bookNow*, 18 – *img_tv*, 19 – *img_projector*, 20 – *img_webCam*, 21 – *img_whiteBoard*, 22 – *img_personBlack*, 23 – *lab_book*, 24 – *btn_15mBookNow*, 25 – *btn_30mBookNow*, 26 – *btn_45mBookNow*, 27 – *btn_1hBookNow*, 28 – *lab_bookPending2*, 29 – *lab_tillBook*, 30 – *lab_tillBookBy*, 31 – *lab_nextBookW*, 32 – *lab_nextBookByW*, 33 – *lab_roomNameW*, 34 – *lab_occupied*, 35 – *lab_personsW*, 36 – *btn_extend*, 37 – *btn_finish*, 38 – *btn_bookNext*, 39 – *img_logoWhite*, 40 – *img_personWhite*, 41 – *lab_bookNextTxt*, 42 – *btn_30mBookNext*, 43 – *btn_45mBookNext*, 44 – *btn_1hBookNext*, 45 – *btn_cancelBookNext*, 46 – *lab_extendTxt*, 47 – *btn_15mExtend*, 48 – *btn_30mExtend*, 49 – *btn_45mExtend*, 50 – *btn_cancelExtend*, 51 – *lab_finishTxt*, 52 – *btn_yesFinish*, 53 – *btn_noFinish*, 54 – *lab_bookNowResult*, 55 – *lab_extendResult*, 56 – *lab_bookNextResult*, 57 – *lab_custScr1_1*, 58 – *lab_custScr1_2*, 59 – *lab_custScr1_3*, 60 – *lab_custScr1_4*, 61 – *lab_custScr1_5*, 62 – *lab_custScr1_6*, 63 – *lab_custScr1_7*, 64 – *lab_custScr1_8*, 65 – *lab_custScr1_9*, 66 – *lab_custScr1_10*, 67 – *but_custScr1_1*, 68 – *but_custScr1_2*, 69 – *but_custScr1_3*, 70 – *but_custScr1_4*, 71 – *but_custScr1_5*, 72 – *but_custScr1_6*, 73 – *pnl_custScr1_1*, 74 – *pnl_custScr1_2*, 75 – *pnl_custScr1_3*, 76 – *pnl_custScr1_4*, 77 – *pnl_custScr1_5*, 78 – *img_custScr1_1*, 79 – *img_custScr1_2*, 80 – *img_custScr1_3*, 81 – *img_custScr1_4*, 82 – *img_custScr1_5*, 83 – *lab_custScr2_1*, 84 – *lab_custScr2_2*, 85 – *lab_custScr2_3*, 86 – *lab_custScr2_4*, 87 – *lab_custScr2_5*, 88 – *lab_custScr2_6*, 89 – *lab_custScr2_7*, 90 – *lab_custScr2_8*, 91 – *lab_custScr2_9*, 92 – *lab_custScr2_10*, 93 – *but_custScr2_1*, 94 – *but_custScr2_2*, 95 – *but_custScr2_3*, 96 – *but_custScr2_4*, 97 – *but_custScr2_5*, 98 – *but_custScr2_6*, 99 – *pnl_custScr2_1*, 100 – *pnl_custScr2_2*, 101 – *pnl_custScr2_3*, 102 – *pnl_custScr2_4*, 103 – *pnl_custScr2_5*, 104 – *img_custScr2_1*, 105 – *img_custScr2_2*, 106 – *img_custScr2_3*, 107 – *img_custScr2_4*, 108 –

img_custScr2_5, 109 – *lab_custScr3_1*, 110 – *lab_custScr3_2*, 111 – *lab_custScr3_3*, 112 – *lab_custScr3_4*, 113 – *lab_custScr3_5*, 114 – *lab_custScr3_6*, 115 – *lab_custScr3_7*, 116 – *lab_custScr3_8*, 117 – *lab_custScr3_9*, 118 – *lab_custScr3_10*, 119 – *but_custScr3_1*, 120 – *but_custScr3_2*, 121 – *but_custScr3_3*, 122 – *but_custScr3_4*, 123 – *but_custScr3_5*, 124 – *but_custScr3_6*, 125 – *pnl_custScr3_1*, 126 – *pnl_custScr3_2*, 127 – *pnl_custScr3_3*, 128 – *pnl_custScr3_4*, 129 – *pnl_custScr3_5*, 130 – *img_custScr3_1*, 131 – *img_custScr3_2*, 132 – *img_custScr3_3*, 133 – *img_custScr3_4*, 134 – *img_custScr3_5*, 135 – *lab_custScr4_1*, 136 – *lab_custScr4_2*, 137 – *lab_custScr4_3*, 138 – *lab_custScr4_4*, 139 – *lab_custScr4_5*, 140 – *lab_custScr4_6*, 141 – *lab_custScr4_7*, 142 – *lab_custScr4_8*, 143 – *lab_custScr4_9*, 144 – *lab_custScr4_10*, 145 – *but_custScr4_1*, 146 – *but_custScr4_2*, 147 – *but_custScr4_3*, 148 – *but_custScr4_4*, 149 – *but_custScr4_5*, 150 – *but_custScr4_6*, 151 – *pnl_custScr4_1*, 152 – *pnl_custScr4_2*, 153 – *pnl_custScr4_3*, 154 – *pnl_custScr4_4*, 155 – *pnl_custScr4_5*, 156 – *img_custScr4_1*, 157 – *img_custScr4_2*, 158 – *img_custScr4_3*, 159 – *img_custScr4_4*, 160 – *img_custScr4_5* (update screen element message with any other value for B_1 is ignoring by tablet).

Bytes starting from B_2 are specific for different update types.

In format 1 - update label text message (B_0 is set to 1) Byte 2 (B_2) is size of the new text string N bytes for the label (Figure 3-7). The size (B_2) is permitted to have values:

- 0x00 – label change its value with space symbol.
- 0x01 – 0x63 – size of the text string for label (maximum size is 99 bytes; if value is more then 99 (0x63 in Hexadecimal) and not 0xFF tablet ignore string after 99th byte).
- 0xFF – label text reset to default value that set in elements.yml configuration file.

Subsequent bytes in update label text message are new text string of N bytes for label (Fig. 3-7).

In format 2 - update screen element visibility message (B_0 is set to 2) Byte 2 (B_2) is new visibility state (any other values are ignored by tablet):

- 0x00 – screen element switches to invisible state.
- 0x01 – screen element switches to visible state.

In format 3 - update multiple screen element, F_1 to F_n refers to the format 1 or format 2, depending on what the update type is. It is however not required to repeat the first bytes (message ID) for each of the update formats (F_1 to F_n). With the first bytes omitted, each of F_1 to F_n looks like the following blocks

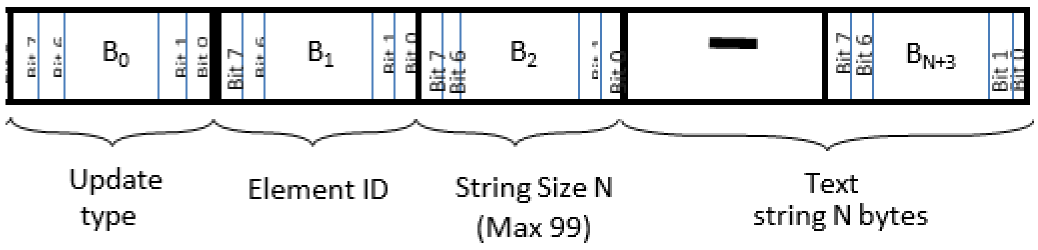


Figure 13: F1 to Fn for format 3 – update text

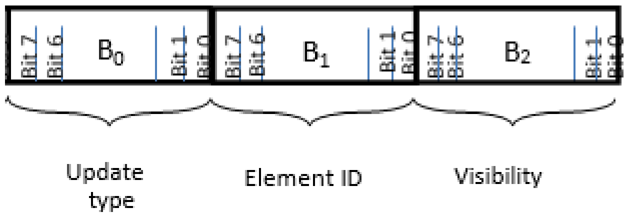


Figure 14: F1 to Fn for format 3 - update visibility

The following table describes the update screen element downlink acknowledgment.

Table 3-12: DL Frame Update screen element acknowledgement

Information Type	**Example Message ID Field Value**	**Maximum DL Frame size (Bytes)**	**Information**
Update screen element message (example *header_id* is 0x45)	0xC5	103	ACK
	0x45	1	NACK

Example update screen element messages:

- 0x: **C5 01 05 09 54 65 73 74 20 74 65 78 74** – message ID with ACK bit (this message is mandatory for tablet); update type B_0 is 0x01 (update label text); element ID B_1 is 0x05 (0x05 that is 5 in Decimal is *lab_loraStatus* label); string size B_2 is 0x09 (0x09 is 9 in Decimal that means that new text string is consist of 9 symbols); string B_3 – B_{11} are ASCII codes for string "Test text".

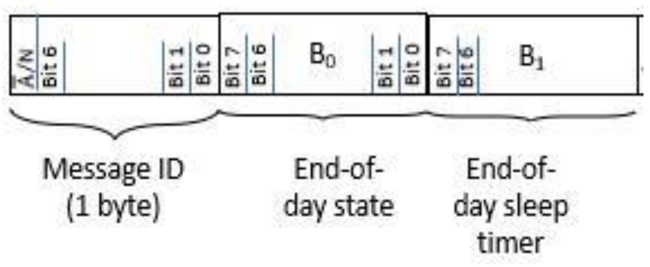
- 0x: C5 02 05 00 – message ID with ACK bit (this message is mandatory for tablet); update type B_0 is 0x02 (update screen element visibility); element ID B_1 is 0x05 (0x05 that is 5 in Decimal is *lab_loraStatus* label); visibility state B_2 is 0x00 (set screen element invisible).

- 0x: **C5 01 05 FF** – message ID with ACK bit (this message is mandatory for tablet); update type B_0 is 0x01 (update label text); element ID B_1 is 0x05 (0x05 that is 5 in Decimal is *lab_loraStatus* label); string size B_2 is 0xFF (set default text that is determined in elements.yml configuration file for label).
- Format 3 example – To update the text in element "3B" and element "3E" to 50 and 90 respectively, simply send the following command: **0x C5 02 01 3B 02 35 30 01 3E 02 39 30**
  - C5: message ID with ACK bit (this is mandatory for tablet)
  - 02: B_0 is number of elements to be updated.
  - 01 3B 02 35 30: F_1
    Where 01: update text option
    3B: element 1 ID
    02: element 1 text length
    35 30: element 1 text converted to Unicode (e.g., "50" is "00350030" in Unicode)
  - 01 3E 02 39 30: F_2
    Where 01: update text option
    3E: element 2 ID
    02: element 2 text length
    39 30: element 2 text element 1 text converted to Unicode (e.g., "90" is "00390030" in Unicode)

3.1.3.11 End-of-day sleep downlink

The following the DL frame formats for end-of-day sleep message.

Figure 15: The format of a DL end-of-day sleep message block.



Message ID bit 7 (A/N) of the message ID determines whether message is ack or nacked. Nacked message is ignored.

End-of-day state Byte 0 (B_0) is new state of end-of-day deep sleep (any other values are ignored by tablet):

- 0x00 – tablet turns off end-of-day deep sleep.
- 0x01 – tablet turns on end-of-day deep sleep.

B_1 can be used to specify the amount of time the tablet will be in EoD sleep for. It accepts a 2-byte unsigned integer value and rejects any other number format. The tablet goes to deep sleep immediately a 0x01 is sent to B_0 and valid value is sent to B_1.

NOTE: For battery powered units after end-of-day sleep message with 0x01 in B_0, the tablet will go to deep sleep mode for either a 14-hour predefined timer if no time was specified in B_1, or for the specified time in B_1. The same message with 0x00 in B_0 is turned on unit comes out of deep sleep. To come out of deep sleep user may tap the EPD screen to wake up but would go back to sleep for predefined timer of 30 seconds. The application may send room status response with EPD_E and TS_E bits turned ON (see 3.1.3.7) before the 14-hour timer expiry to wake up from deep sleep or end-of-day sleep message with 0x00 in B_0, at this point the unit will be out of deep sleep mode and ready for regular operation. To summarize, the unit will stay in deep sleep mode for 14 hours. If user taps the screen in middle of this 14-hour period, unit will wake up for 30s and go back to deep sleep. The application can wake up the unit before the 14-hour period by sending EPD_E and TS_E bits ON in room status response or by sending special End-of-day sleep downlink (see details in 3.1.2.11). Both Room Status and End-of-day sleep message manage end-of-day deep sleep.

The following table describes the end-of-day sleep downlink acknowledgment.

Table 3-13: DL Frame End-of-day sleep acknowledgement.

Information Type	Example Message ID Field Value	Maximum DL Frame size (Bytes)	Information
End-of-day sleep message (example *header_id* is 0x46)	0xC6	4	ACK
	0x46	1	NACK

Example end-of-day sleep messages:

- 0x: C6 00 – message ID with ACK bit (this message is mandatory for tablet); end-of-day state B_0 is 0x00 (turn off end-of-day deep sleep).
- 0x: C6 01 – message ID with ACK bit (this message is mandatory for tablet); end-of-day

state B_0 is 0x01 (turn on end-of-day deep sleep).

- 0x: C6 01 02 1C – message ID with ACK bit (this message is mandatory for tablet); end of day state is 0x01, timer is 0x09. Tablet goes to deep sleep for 540 minutes (9 hours) immediately after this message is received
- 0x: C6 01 00 00 – message ID with ACK bit (this message is mandatory for tablet); end of day state is 0x01, timer is 0x00 00. Tablet ignores this since timer is set to zero.
- 0x: 46 – message ID with NACK bit (this message will be ignored by tablet).

3.1.4 MRDT Sensor Command and Control

Configuration changes are not retained after a power cycle unless they are saved in the flash memory. Figure 3-1 shows the structure of the Command-and-Control registers. In this table, B_i refers to data byte indexed *i* as defined in Figure 3-1.

Table 3-14: Sensor Command & Control Register

Address	Access	Name	# Bytes	Description
0x70	W	Flash Memory Write Command	2	B_0, bit 5: Write App Config B_0, bit 6: Write LoRa Config B_1, bit 0: Restart Sensor In all cases: 0 = De-asserted, 1 = Asserted Other bits are ignored.
0x71	R	FW Version	7	B_0: App version major B_1: App version minor B_2: App version revision B_3: LoRaMAC version major B_4: LoRaMAC version minor B_5: LoRaMAC version revision B_6: LoRaMAC region number
0x72	W	Reset Config Registers to Factory Defaults[2]	1	0x0A: Reset App Config 0xB0: Reset LoRa Config 0xBA: Reset both App and LoRa Configs Any other value is ignored.

[2] Resetting to factory defaults takes effect on the next power cycle.

Note: The Flash Memory Write Command is always executed after the full DL configuration message has been decoded. The reset command should always be sent as an “unconfirmed” DL message. Failure to do so may cause the NS to continually reboot the Sensor.

3.1.4.1 LoRaMAC Region

The LoRaMAC region is indicated by B_6 in the FW Version register (Reg 0x71). Current LoRaMAC regions and corresponding region numbers are listed in Table 3-15

Table 3-15: LoRaMAC Regions and Region Numbers

LoRaMAC Region	**Region Number**
EU868	0
NA915	1
AS923	2
AU915	3
IN865	4
CN470	5
KR920	6
RU864	7

3.1.4.2 Command Examples

In the following examples, the Command Field is boldfaced:

- Write application configuration to flash memory payload: {0x**F0** 20 00}
- Write application and LoRa configurations to flash memory
    - o DL payload: {0x **F0** 60 00}
- Reboot Device
    - o DL payload: {0x **F0** 00 01}
- Read FW versions, and reset application configuration to factory defaults
    - o DL payload: {0x **71 F2** 0A}

3.1.5 Preventing Sensor Bricking

Care has been taken to avoid stranding (hard or soft bricking) the Sensor during reconfiguration. Hard bricking refers to the condition that the Sensor does not transmit any more as all periodic and event-based reporting (see subsequent sections) have been disabled and the configuration has been saved to the Flash memory. Soft bricking refers to the condition where the Sensor has been configured such that all event-based reporting is disabled and any periodic reporting is either disabled or has a period of larger than a week. Therefore, transmissions from a soft-bricked Sensor cannot be smaller than a week apart.

To avoid these situations, for any reconfiguration command sent to the Sensor, the following algorithm is automatically executed:

After the reconfiguration is applied, if all event-based reporting is disabled, then periodic reporting is checked (see Section 3.1.3.1 for periodic reporting). If all periodic reporting is disabled or the minimum non-zero period is greater than a week, then to avoid bricking the Sensor, the core tick is set to 86400 (i.e., one day), and the battery voltage tick is set to 1 (one).

References

[1] LoRa Alliance, "LoRaWAN Specification," ver. 1.0.3, Jul 2018.

[2] LoRa Alliance, "LoRaWAN Regional Parameters," ver. 1.1, rev. B, Jan 2018.

[3] State Machine, "Meeting Room Tablet State Machine," ver 0.4, April 17, 2020