enLink

enLink - Download Settings Payload

Downlink payloads are sent to re-configure the device. After the payload is data is processed the device acknowledges the message with an ACK/NACK and the identifier code.

Payload Structure

Header Byte (0xA5)	Data Length (n)	Settings Data
1 byte	1 byte	n bytes

Settings

Identifier / Command	Name	Data Length	Value Size	Value	Reboot Required?
0xFF	Reboot	1	(Zero)		
0x00	(Reserved)				
0x01	(Reserved)				
0x02	Public Network	2	1 byte	0x00 / 0x01	Yes
0x01	(Reserved)				
0x01	(Reserved)				
0x05	AppEUI	9	8 bytes		Yes
0x06	AppKey	17	16 bytes		Yes
0x07	Auto Data Rate (ADR)	2	1 byte	0x00 / 0x01	
0x08	Duty Cycle	2	1 byte	0x00 / 0x01	
0x09	Message Confirmation	2	1 byte	0x00 / 0x01	
0x0A	Transmit/Receive Port	2	1 byte	1 to 255, excluding 224	
0x0B	Default Data Rate Index	2	1 byte	0x01 – 0x06	
0x0C	Transmit Interval Index	2	1 byte	0x01 – 0x0A	
0x0D	Transmit Power Index	2	1 byte	0x01 – 0x06	

The following are used only in the AQM, Zone and ZonePlus (with Light Sensor)

0x20	Lux Scale Parameter	3	2 bytes	0.0 to 65.535	
0x21	Lux Offset Parameter	3	2 bytes	0 to 65535	

The following are used only in the AQM

0x22	Case Fan Run Time	3	2 bytes	10 to 600 seconds	
0x23	Particulate Fan Run Time	3	2 bytes	10 to 60 seconds	

The following are used only in enLink devices with the Sunrise CO2 Sensor

0x24	Enable/Disable Auto-Calibration	2	1 byte	0x00 / 0x01	
0x25	Set Target CO2 Level	3	2 bytes	100 to 1000 ppm	
0x26	Set to Known CO2 Level	3	2 bytes	10 to 2000 ppm	
0x27	Reset to Factory Calibration	1	(Zero)		
0x28	Set Auto-Cal Interval	3	2 bytes	24 to 8760 hours	

For simple 1 byte Enable/Disable (On/Off) values use 0x00=Disable (Off) and 0x01=Enable (On)

Reboot Example: A5 01 FF

Header Byte (0xA5)	Data Length (n)	Identifier Byte
0xA5	0x01	0xFF

Enable Message Confirmation Example: A5 02 09 01

Header Byte (0xA5)	Data Length (n)	Identifier Byte	Value
0xA5	0x02	0x09	0x01

Return code Example: ACK – A5 06 09
Successfully changed the Message Confirmation Option

Header Byte (0xA5)	Result (ACK 0x06)	Identifier Byte
0xA5	0x06	0x09

Return code: NACK – A5 15 0A
Failed to change the transmit/receive port

Header Byte (0xA5)	Result (NACK 0x15)	Identifier Byte
0xA5	0x15	0x0A

Index Tables
Data Rate / Spreading Factor / Bandwidth

Index	EU868
0	DR0 SF12 BW125
1	DR1 SF11 BW125
2	DR2 SF10 BW125
3	DR3 SF9 BW125
4	DR4 SF8 BW125
5	DR5 SF7 BW125

Index	US915 (Hybrid)
0	DR0 SF10 BW125
1	DR1 SF9 BW125
2	DR2 SF8 BW125
3	DR3 SF7 BW125
4	DR4 SF8 BW500

Index	Transmit Interval	Message
1	30 s	A5 02 0C 01
2	1 min	A5 02 0C 02
3	2 min	A5 02 0C 03
4	5 min	A5 02 0C 04
5	10 min	A5 02 0C 05
6	15 min	A5 02 0C 06
7	20 min	A5 02 0C 07
8	30 min	A5 02 0C 08
9	1 hour	A5 02 0C 09
10	2 hours	A5 02 0C 0A
11	3 hours	A5 02 0C 0B

Transmit Power

Index	EU 868
1	16 dBm
2	14 dBm
3	11 dBm
4	9 dBm
5	8 dBm
6	6 dBm
7	4 dBm
8	2 dBm

Index	US195 (Hybrid)
6	20 dBm
7	18 dBm
8	16 dBm
9	14 dBm
10	12 dBm
11	10 dBm

Index 1 to 5 are not used

Lux Sensor Scaling

To scale the lux reading to compensate for the enclosure light pipe, a scaling factor is applied to the sensor value:

Adjusted_Reading = (Sensor_Value x Scale) + Offset

Defaults are: Scale = 2.0 and Offset = 0

For example, set Scale to 12.345 (12345 in hexadecimal is 0x3039)
Message is: **A5 03 20 30 39**

Setting CO2 Examples

To Enable Auto-Calibration:
Message is: **A5 02 24 01**

To set the auto-calibration target to 450ppm
Message is: **A5 03 25 01 C2**

To set the sensor to known CO2 concentration of 780ppm (0x030C)
Message is: **A5 03 26 03 0C**

To reset the sensor back to factory calibration
Message is: **A5 01 27**

To set the auto-calibration interval to 10 days (240 hours, 0x00F0)
Message is: **A5 03 28 00 F0**

```
// Used for decoding enLink Messages from Downlink Replies
// DN 01 Feb 2021

if (!msg.eui) return null;

// ------------------------------------------------------------------------------------------
// Ignore Port 0 Possible MAC Command
if (msg.port === 0) {
    return null;
}
// Ignore zero payloads
if (msg.payload) {
    if (msg.payload.length === 0) {
        return null;
    }
} else {
    return null;
}
// ------------------------------------------------------------------------------------------

// V1 - Downlink reply message Header and ACK/NAK
const ENLINK_HEADER = 0xA5;
const ENLINK_ACK = 0x06;
const ENLINK_NACK = 0x15;
// Downlink reply message values
const ENLINK_SET_PUBLIC = 0x02;
const ENLINK_SET_USE_USER_DEVEUI = 0x03;
const ENLINK_SET_USER_DEVEUI = 0x04;   // 8 bytes
const ENLINK_SET_APPEUI = 0x05;   // 8 bytes
const ENLINK_SET_APPKEY = 0x06;   // 16 bytes
const ENLINK_SET_ADR = 0x07;
const ENLINK_SET_DUTY_CYCLE = 0x08;
const ENLINK_SET_MSG_ACK = 0x09;
const ENLINK_SET_PORT = 0x0A;
const ENLINK_SET_DR_INDEX = 0x0B;   // Data Rate Index 0~6
const ENLINK_SET_TX_INDEX = 0x0C;   // Data Rate Index 0~10
const ENLINK_SET_POW_INDEX = 0x0D;   // Data Rate Index 0~6
const ENLINK_SET_LUX_SCALE = 0x20;
const ENLINK_SET_LUX_OFFSET = 0x21;

const ENLINK_SET_CASE_FAN_RUN_TIME = 0x22;
const ENLINK_SET_HPM_FAN_RUN_TIME = 0x23;

const ENLINK_SET_CO2_ABC_ENABLE = 0x24;
const ENLINK_SET_CO2_ABC_TARGET_PPM = 0x25;
const ENLINK_SET_CO2_KNOWN_PPM = 0x26;
const ENLINK_SET_CO2_FACTORY_CALIB = 0x27;
const ENLINK_SET_CO2_ABC_INTERVAL = 0x28;

const ENLINK_REBOOT = 0xFF;

// Function to decode enLink Messages
function DecodePayload(data) {
    var obj = {};
    obj.v1_msg_eui = msg.eui.slice(-8);
    var msg_ok = false;
    for (i = 0; i < data.length; i++) {
        console.log(data[i]);
        switch (data[i]) {
            // Parse Reply Message
            case ENLINK_HEADER: // Response from enLink Device
                if (data[i + 1] == ENLINK_ACK) {
                    obj.reply = "ACK";
                    msg_ok = true;
                } else if (data[i + 1] == ENLINK_NACK) {
                    obj.reply = "NACK";
                    msg_ok = true;
                } else {
                    obj.reply = "Reply parse failure";
                }

                if (data[i + 2] == ENLINK_SET_PUBLIC) {
                    obj.command = "Set Public";
                } else if (data[i + 2] == ENLINK_SET_USE_USER_DEVEUI) {
                    obj.command = "Set Use User DevEUI";
                } else if (data[i + 2] == ENLINK_SET_USER_DEVEUI) {
                    obj.command = "Set User DevEUI";
                } else if (data[i + 2] == ENLINK_SET_APPEUI) {
                    obj.command = "Set AppEUI";
                } else if (data[i + 2] == ENLINK_SET_APPKEY) {
                    obj.command = "Set AppKEY";
                } else if (data[i + 2] == ENLINK_SET_ADR) {
                    obj.command = "Set ADR";
                } else if (data[i + 2] == ENLINK_SET_DUTY_CYCLE) {
                    obj.command = "Set Duty Cycle";
                } else if (data[i + 2] == ENLINK_SET_MSG_ACK) {
                    obj.command = "Set Message Confirmation";
                } else if (data[i + 2] == ENLINK_SET_PORT) {
```

```
                obj.command = "Set Port";
            } else if (data[i + 2] == ENLINK_SET_DR_INDEX) {
                obj.command = "Set Data Rate";
            } else if (data[i + 2] == ENLINK_SET_TX_INDEX) {
                obj.command = "Set TX Interval";
            } else if (data[i + 2] == ENLINK_SET_POW_INDEX) {
                obj.command = "Set TX Power";
            } else if (data[i + 2] == ENLINK_SET_LUX_SCALE) {
                obj.command = "Set LUX Scale";
            } else if (data[i + 2] == ENLINK_SET_LUX_OFFSET) {
                obj.command = "Set LUX Offset";

            } else if (data[i + 2] == ENLINK_SET_CASE_FAN_RUN_TIME) {
                obj.command = "Set Case Fan Run Time";
            } else if (data[i + 2] == ENLINK_SET_HPM_FAN_RUN_TIME) {
                obj.command = "Set Particle Sensor Fan Run Time";

            } else if (data[i + 2] == ENLINK_SET_CO2_ABC_ENABLE) {
                obj.command = "Set CO2 Sensor Auto-Calib Enable/Disable Flag";
            } else if (data[i + 2] == ENLINK_SET_CO2_ABC_TARGET_PPM) {
                obj.command = "Set Co2 Sensor ABC Target";
            } else if (data[i + 2] == ENLINK_SET_CO2_KNOWN_PPM) {
                obj.command = "Set CO2 Sensor to Known ppm";
            } else if (data[i + 2] == ENLINK_SET_CO2_FACTORY_CALIB) {
                obj.command = "Set CO2 Sensor to Factory Calib";
            } else if (data[i + 2] == ENLINK_SET_CO2_ABC_INTERVAL) {
                obj.command = "Set CO2 Sensor Auto-Calib Interval";

            } else if (data[i + 2] == ENLINK_REBOOT) {
                obj.command = "Reboot";
            } else {
                obj.command = "Command parse failure: " + data[i + 2];
            }

            i = data.length;
            break;

        default: // something is wrong with data
            i = data.length;
            break;
    }
  }
  if (msg_ok) {
    return obj;
  } else {
    return null;
  }
}

var res = DecodePayload(msg.payload);
if (res !== null) {
  var json = JSON.stringify(res, null, 4);
  msg.payload = json;
  return msg;
} else {
  return null;
}
```

JavaScript to decode enLink reply messages for Node-RED and display values in the debug pane

Document Revision History

Date	Revision	Changes
1 Oct 2017	1.0	• Initial Document
31 Dec 2017	1.1	• Corrected Examples • Added extra descriptor byte to ACK/NACK messages • Firmware 1.1
2 Feb 2018	1.2	• Add LUX sensor configuration parameters for offset and scale • Firmware 1.4
1 Jun 2018	1.3	• Covers all enLink end-node models • Changed Data Rate indexes to start from zero, to match configuration menu • Add Fan run time configuration parameters for Case Fan and Particulate Sensor Fan • Added EU868 and US915(Hybrid) data rate and transmit power settings • Firmware 1.12
1 Feb 2021	4.28	• Change document version to match enLink Firmware Version • Added JS decoder for Node-RED • Add CO2 Sensor commands for: ○ Enable/Disable Auto-Calibration ○ Set Target CO2 Level ○ Set to Known CO2 Level ○ Reset to Factory Calibration ○ Set Auto-Cal Interval (Regular Interval)