Sample Decoder for Payload of LoRaWAN Sensor

WSLRW-MN-EN	APR-2020



Sample Decoder for Payload of LoRaWAN Sensor

THIS IS **OBSOLETE** MANUAL

Please access https://www.iot.daviteq.com/wireless-sensors for updated manual

1. Decoder sample

ℹ You can download **HERE**

```
/* ********************************************
* bytes to number
* https://docs.microsoft.com/en-us/cpp/cpp/data-type-ranges?view=msvc-160
**********************************************/
function bytesToFloat(bytes) {
// Reference link:https://stackoverflow.com/a/37471538 by Ilya Bursov
var bits = bytes[0]<<24 | bytes[1]<<16 | bytes[2]<<8 | bytes[3];
var sign = (bits>>>31 === 0) ? 1.0 : -1.0;
var e = bits>>>23 & 0xff;
var m = (e === 0) ? (bits & 0x7fffff)<<1 : (bits & 0x7fffff) | 0x800000;
var f = sign * m * Math.pow(2, e - 150);
return f;
}
function bytesToUnsignedInt32(bytes) {
var v = bytes[0] << 24 | bytes[1] << 16 | bytes[2] << 8 | bytes[3];
return v;
}
function bytesToInt32(bytes) {
// 0...31: representation bits
// 31: sign bit
var v = bytes[0] << 24 | bytes[1] << 16 | bytes[2] << 8 | bytes[3];
return v > 0x7FFFFFFF ? v - 0x0100000000 : v;
}
function bytesToUnsignedInt8(bytes) {
var v = bytes[0];
return v;
}
function bytesToInt8(bytes) {
// 0...7: representation bits
// 8: sign bit
var v = bytes[0];
```

```
return v > 0x7f ? v - 0x0100 : v;
}
function bytesToUnsignedInt16(bytes) {
var v = bytes[0] << 8 | bytes[1];
return v;
}
function bytesToInt16(bytes) {
// 0...14: representation bits
// 15: sign bit
var v = (bytes[0] << 8) | bytes[1];
return v > 0x7fff ? v - 0x010000 : v;
}
/**----------------------------------**/

/**------GLOBIOTS SENSOR TYPE-------**/
var I2C_ATE_SENSOR_TYPE = 0x01;
var I2C_ATH_SENSOR_TYPE = 0x02;
var I2C_PPS_SENSOR_TYPE = 0x04;
var SMT01_SENSOR_TYPE = 0x0C;
var LIQUID_LEVEL_SENSOR_TYPE = 0x0E;
var I2C_TILT_SENSOR_TYPE = 0x0D;
/**----------------------------------**/
var STATUS_BATTERY_BIT_MASK = 0xC0;
var STATUS_ERROR_BIT_MASK = 0x30;
var STATUS_ALARM_1_BIT_MASK = 0x0C;
var STATUS_ALARM_2_BIT_MASK = 0x03;
var STATUS_ALARM_3_BIT_MASK = 0x03;

var GLOBIOTS_DEFINED_DATA = {};
var listParameterMapping = [];
var sensorType = bytes[0];
var batteryLevel = (bytes[1] & STATUS_BATTERY_BIT_MASK) >> 6;
var sensorError = (bytes[1] & STATUS_ERROR_BIT_MASK) >> 4;
var parameter1Alarm = (bytes[1] & STATUS_ALARM_1_BIT_MASK) >> 2; //1st - Parameter alarm status
var parameter2Alarm = bytes[1] & STATUS_ALARM_2_BIT_MASK; //2nd - Parameter alarm status


/**------------------------GLOBIOTS MAPPING DATA------------------------**/
var parameterMapping = {};
parameterMapping["PARAMETER_6000"] = batteryLevel;
parameterMapping["PARAMETER_6006"] = sensorError;
parameterMapping["PARAMETER_6007"] = sensorType;

if(sensorType === I2C_ATE_SENSOR_TYPE){
var temperature = bytesToFloat(bytes.slice(2, 6)); //1st - Parameter

parameterMapping["PARAMETER_6002"] = temperature;
parameterMapping["ALARM_6002"] = parameter1Alarm;
} else if(sensorType === I2C_ATH_SENSOR_TYPE){
var humidity = bytesToFloat(bytes.slice(2, 6)); //1st - Parameter
var temperature = bytesToFloat(bytes.slice(6, 11)); //2nd - Parameter

parameterMapping["PARAMETER_6002"] = humidity;
parameterMapping["PARAMETER_6008"] = temperature;
parameterMapping["ALARM_6002"] = parameter1Alarm;
parameterMapping["ALARM_6008"] = parameter2Alarm;
} else if(sensorType === I2C_PPS_SENSOR_TYPE){
var processPressure = bytesToFloat(bytes.slice(2, 6)); //1st - Parameter
var temperature = bytesToFloat(bytes.slice(6, 11)); //2nd - Parameter

parameterMapping["PARAMETER_6002"] = processPressure;
parameterMapping["PARAMETER_6008"] = temperature;
parameterMapping["ALARM_6002"] = parameter1Alarm;
parameterMapping["ALARM_6008"] = parameter2Alarm;
} else if(sensorType === SMT01_SENSOR_TYPE){
var soilMoisture = bytesToFloat(bytes.slice(2, 4)); //1st - Parameter
var temperature = bytesToFloat(bytes.slice(4, 6)); //2nd - Parameter

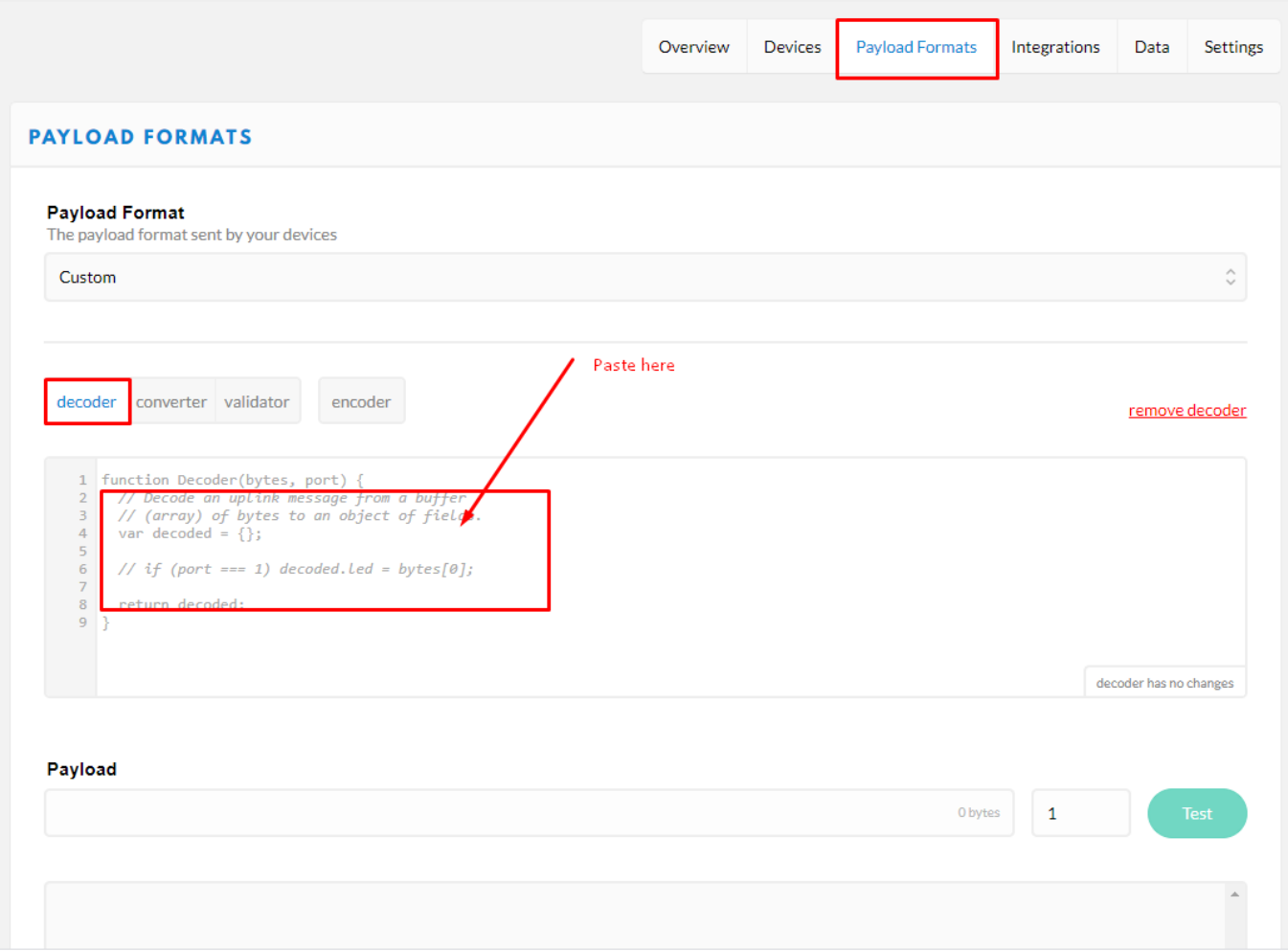
parameterMapping["PARAMETER_6002"] = soilMoisture;
parameterMapping["PARAMETER_6008"] = temperature;
parameterMapping["ALARM_6002"] = parameter1Alarm;
parameterMapping["ALARM_6008"] = parameter2Alarm;
} else if(sensorType === LIQUID_LEVEL_SENSOR_TYPE){
var level = bytesToUnsignedInt16(bytes.slice(2, 4)); //1st - Parameter
var distance = bytesToUnsignedInt16(bytes.slice(4, 6)); //1st - Parameter

parameterMapping["PARAMETER_6002"] = level;
parameterMapping["PARAMETER_6008"] = distance;
parameterMapping["ALARM_6002"] = parameter1Alarm;
parameterMapping["ALARM_6008"] = parameter2Alarm;
} else if(sensorType === I2C_TILT_SENSOR_TYPE){
```

```
var parameter3Alarm = bytes[2] & STATUS_ALARM_3_BIT_MASK;
var realX = bytesToInt16(bytes.slice(3, 5)) / 10 ; //1st - Parameter
var realY = bytesToInt16(bytes.slice(5, 7)) / 10; //2nd - Parameter
var realZ = bytesToInt16(bytes.slice(7, 9)) / 10; //3rd - Parameter
parameterMapping["PARAMETER_6002"] = realX;
parameterMapping["PARAMETER_6008"] = realY;
parameterMapping["PARAMETER_6ACD"] = realZ;
parameterMapping["ALARM_6002"] = parameter1Alarm;
parameterMapping["ALARM_6008"] = parameter2Alarm;
parameterMapping["ALARM_6ACD"] = parameter3Alarm;
}

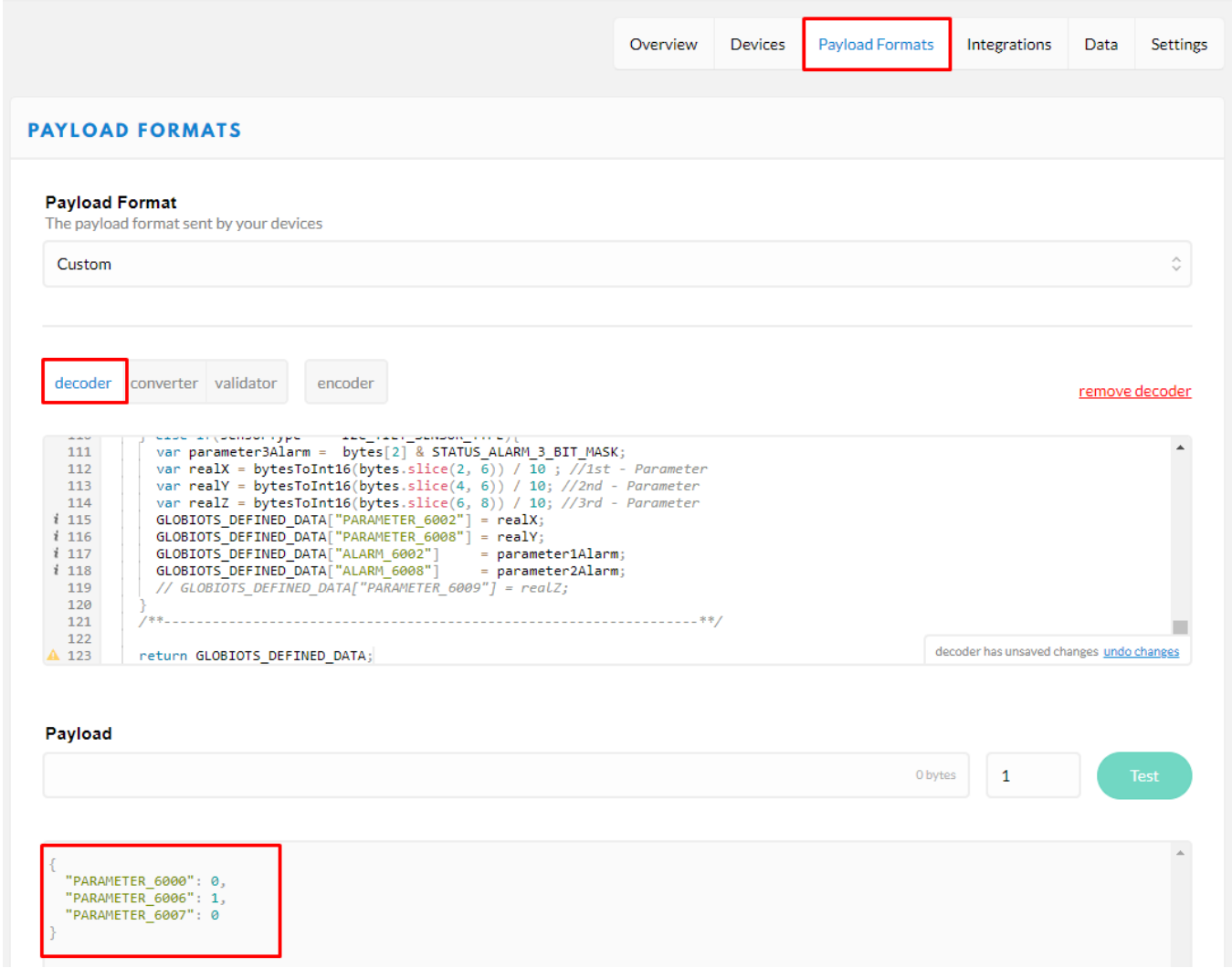
listParameterMapping.push(parameterMapping);
/**-----------------------------------------------------------------**/
GLOBIOTS_DEFINED_DATA['parameterMappings'] = listParameterMapping;
return GLOBIOTS_DEFINED_DATA;
```

2. Place to write decoder



3. Test decoder

Write in the payload you need to decode and then run the test



4. Support contacts

Manufacturer

Daviteq Technologies Inc
No.11 Street 2G, Nam Hung Vuong Res., An Lac Ward, Binh Tan Dist., Ho Chi Minh City, Vietnam.
Tel: +84-28-6268.2523/4 (ext.122)
Email: info@daviteq.com | www.daviteq.com

Distributor in **Australia** and **New Zealand**

Templogger Pty Ltd
Tel: 1800 LOGGER
Email: contact@templogger.net

Revision #4
★ Created Sun, Apr 18, 2021 11:14 PM by Kiệt Anh Nguyễn
Updated Wed, Jan 24, 2024 6:50 AM by Phi Hoang Tran