IoT over LoRaWAN/Sigfox

Supported device types and versions
Communication line configuration
Communication line parameters
Communication station configuration
Station parameters
I/O tag configuration
Literature
Document revisions

Supported device types and versions

The protocol allows communication with devices communicating using the LoRaWAN and Sigfox protocols.

The LoRaWAN protocol is a protocol designed for occasional, energy-efficient wireless transmission of a small amount of data called **payload** (typically several bytes) over long distances between linked objects - in LoRaWAN terminology called *mote* (typically battery-powered sensors) and LoRaWAN gateway.
The gateway can then communicate directly with the D2000 KOM process or send the payload to the network servers or to the cloud, where data processing is performed (deduplication, filtration), followed by the data sent to the D2000 KOM process. In any case, the payload is packed in an **envelope** (e. g. such as a field in a JSON message or in a CSV file) and transferred up to the D2000 KOM process. The payload is decoded there (using Base64 encoding or Base64 + Base16 encoding) and processed.
Payload processing is dependent on device type (defined in the Device Type protocol parameter).

The communication was tested between sensors and LoRaWAN gateway Kerlink IoT Station 868. It was, depending on the installed firmware:

- sending data directly to the D2000 KOM process (JSON messages in UDP packets)
- sending data to the cloud TheThings.Network, from where it was read by D2000 KOM process (JSON messages in MQTT protocol via TCP connection)
- sending data to the cloud Loriot.io, which was forwarding them via MQTT to iot.eclipse.org, from where it was read by D2000 KOM process (JSON messages in MQTT protocol via TCP connection)
- sending data to the cloud Slovanet (loralink.slovanet.sk), from where it was read by the D2000 KOM process (JSON messages in MQTT protocol via TCP connection)

The Sigfox protocol is similar to LoRaWAN, however, it uses different frequency bands. It uses the infrastructure built by Sigfox and its partners and messages (with a maximum payload size of 12 bytes, and a maximum frequency of 140 messages per day) are transmitted to the cloud (backend.sigfox.com) from which they can be obtained through the REST API.

The communication was tested between sensors and the Sigfox cloud using curl utility to retrieve data via HTTPS connection.

Communication line configuration

Configuration for LoRaWAN protocol:

- Communication line category: SerialOverUDP Device Redundant, TCP/IP-TCP, File I/O.
- Configuration of SerialOverUDP Device Redundant line:
  - Local port: port, where the D2000 KOM process receives UDP packets
  - Primary / Backup Device: IP address LoRaWAN gateway
    (pre Connection Type=*Kerlink IoT Station SPN*)
  - Port: port, where the LoRaWAN gateway receives UDP packets (currently unused, as there is no implemented record)
- Configuration of TCP/IP-TCP line:
  - Host: IP address of the server, to which the D200 KOM process connects or redundant addresses separated by a comma or semicolon (for Connection Type=*MQTT client*)
  - Port: server port, to which the D200 KOM process connects

Configuration for Sigfox protocol:

- Configuration of File/IO line:
  - Input file: name of the directory where data (JSON files) downloaded from the Sigfox cloud will be located. A symbolic constant **#APPDIR#**, which specifies application directory, may be used, e.g. *#APPDIR#/Sigfox_input* (valid syntax for Windows and Linux).
  - Archive: name of the directory where data files will be moved after being processed. The D2000 KOM process will move the data files after processing to the subdirectories of this directory, which will be named according to respective stations (e.g B.Sigfox_1) and it will add a timestamp indicating the processing time (e.g. a file msg_21FDA7.txt may be archived as msg_21FDA7_2018-10-26 -06-26-08.txt).

    Similarly to the input file, a symbolic constant **#APPDIR#** may be used, e.g. *#APPDIR#/Sigfox_archiv*
    Note 1: Invalid files or files for which no station has been identified will be moved to the subdirectory *BAD* after processing.
    Note 2: If the archive directory is not specified, all files will be deleted after processing.

Communication line configuration

Communication line - configuration dialog - **Protocol parameters** tab.
The parameters influence some optional protocol parameters. The following protocol line parameters can be used:

Table 1

Parameter	Description	Unit / size	Default Value
Connection Type	Type of connection between the D2000 KOM process and the other party (LoRaWAN gateway, network server, cloud). Currently supported are: • **Kerlink IoT Station SPN (JSON via UDP packets)**: communication with Kerlink IoT Station with firmware SPN (Small Private Network). Line must be of the SerialOverUDP Device Redundant type. • **MQTT Client (JSON via MQTT)**: communication with a network server or cloud using MQTT protocol. The line must be of the TCP/IP-TCP type. • **Sigfox Client (JSON via HTTPS)**: communication with Sigfox cloud	-	Kerlink IoT Station SPN
Mote Field Name	Name of the field with an identifier of the LoRaWAN device (mote). **Note:** For JSON messages that can be structured, the syntax *level1.level2.level3 ...* is supported e.g. *rx.moteeui* and if they contain fields (indexed from 1) then also the syntax *level1[index1].level2[index2].level3 ...* is supported e.g. *rx.gwrx[1].time*. As examples, see the description of I/O tags of the Envelope type. **Note:** For Sigfox data files, which may contain multiple values (historical), index 0 resp. empty index (e.g. *data[].device*) can be used. This indicates that all field elements are to be processed.	-	rx.moteeui
Payload Field Name	Name of the field with the payload. See the notes next to the Mote Field Name parameter.	-	rx.userdata.payload
Payload Encoding	A method of payload encoding in the message. Supported encoding: • **Base16 + Base64 encoding (Kerlink SPN)** - for Connection Type=*Kerlink IoT Station SPN* • **Base64 encoding (TheThings.network)** - for Connection Type=*MQTT Client* communicating with TheThings.network cloud • **Base64 encoding (Loriot, Slovanet, Sigfox)** - for Connection Type=*MQTT Client* communicating with LoRaWAN clouds (Lorio, Slovanet) as well as Sigfox • **None** - message contains a payload without encoding - not yet used	-	Base16 + Base64 encoding
Time Field Name	Name of the field with a timestamp. If the field is not found, the current time is assigned to the values. See the notes next to the Mote Field Name parameter.	-	rx.gwrx[1].time
Time Mask	Mask for parsing a value in the field with a timestamp. **Note:** from settings of time station parameters depends whether the time is interpreted as local or UTC with configured offset. Special masks are: • **UNIX** - the numeric value represents the number of seconds from epoch 00:00:00 01.01.1970 UTC. • **UNIXMS** - the numeric value represents the number of milliseconds from epoch 00:00:00.000 01.01.1970 UTC.	-	yyyy-mm-dd hh:mi:ss
Frame Type Field Name	The name of the field indicating the message type. If the value is empty, the message type is not distinguished. (For example, cloud Loriot sends messages of various types.)	-	
Frame Type Field Required Value	If the message type differentiation is active (non-empty value of Frame Type Field Name parameter), the message type must match the specified value, otherwise, the message is ignored.	-	
Full Debug	Writing detailed information about sending and receiving values in a log file.	YES /NO	NO
Parameters specific for Connection Type=*MQTT Client*.			
MQTT User Name	See the description of the User Name parameter in the MQTT protocol documentation.		
MQTT Password	See the description of the Password parameter in the MQTT protocol documentation.		
MQTT Topic Filter	See the description of the Topic Filter parameter in the MQTT protocol documentation.		+/+/+/up
MQTT Subscribe QoS	See the description of the Subscribe QoS parameter in the MQTT protocol documentation.		
MQTT Client ID	See the description of the Client ID parameter in the MQTT protocol documentation.		
MQTT Clean Session Flag	See the description of the Clean Session Flag parameter in the MQTT protocol documentation.		

MQTT Publish Format	Format of JSON message used while writing a value. The content of the I/O tag of Write type will be encoded (depending on the Payload Encoding parameter) and inserted into the message, where it will replace the *#PAY#* string. The default value *"{"port":1, "confirmed":false, "payload_raw":#PAY#}* was tested when sending data to cloud TheThings. Network.	-	"{"port":1, "confirmed": false, "payload_raw": #PAY#}
MQTT Publish QoS	See the description of the Publish QoS parameter in the MQTT protocol documentation.		
MQTT Ping Interval	See the description of the Ping Interval parameter in the MQTT protocol documentation.		
MQTT Reply Timeout	See the description of the Reply Timeout parameter in the MQTT protocol documentation.		
MQTT Wait Timeout	See the description of the Wait Timeout parameter in the MQTT protocol documentation.		
MQTT Max. Wait Retry	See the description of the Max. Wait Retry parameter in the MQTT protocol documentation.		

Line parameters tested for Connection Type=*Kerlink IoT Station SPN* for Kerlink IoT Station 868 with firmware SPN

Parameter	Value
Connection Type	Kerlink IoT Station SPN
Mote Field Name	rx.moteeui
Payload Field Name	rx.userdata.payload
Payload Encoding	Base16 + Base64 encoding
Time Field Name	rx.gwrx[1].time
Time Mask	yyyy-mm-dd hh:mi:ss
Frame Type Field Name	
Frame Type Field Required Value	

Line parameters tested for Connection Type=*MQTT client* for TheThings.network

Parameter	Value
Connection Type	MQTT client
Mote Field Name	dev_id or hardware_serial
Payload Field Name	payload_raw
Payload Encoding	Base64 encoding
Time Field Name	metadata.time
Time Mask	yyyy-mm-dd hh:mi:ss.mss
Frame Type Field Name	
Frame Type Field Required Value	
MQTT User Name	ipesoft-test
MQTT Password	***
MQTT Topic Filter	+/+/+/up
MQTT Client ID	D2000kom
MQTT Clean Session Flag	NO
MQTT Publish Format	{"port":1, "confirmed":false, "payload_raw":#PAY#}
MQTT Publish QoS	QoS_0, QoS_1, QoS_2
MQTT Ping Interval	60
MQTT Reply Timeout	20
MQTT Wait Timeout	00.100

MQTT Max. Wait Retry	3

Line parameters tested for Connection Type=*MQTT client* for Loriot.io with the following setup:

- Output via protocol MQTT
- MQTT broker: iot.eclipse.org
- MQTT topic: com/ipesoft/iot

Parameter	Value
Connection Type	MQTT client
Mote Field Name	EUI
Payload Field Name	data
Payload Encoding	Base16 encoding
Time Field Name	ts
Time Mask	UNIXMS
Frame Type Field Name	cmd
Frame Type Field Required Value	rx
MQTT User Name	
MQTT Password	
MQTT Topic Filter	com/ipesoft/iot
MQTT Client ID	D2000kom
MQTT Clean Session Flag	NO
MQTT Publish Format	
MQTT Publish QoS	QoS_1
MQTT Ping Interval	60
MQTT Reply Timeout	20
MQTT Wait Timeout	00.100
MQTT Max. Wait Retry	3

Line parameters tested for Connection Type=*MQTT client* towards LoraLINK Slovanet:

Parameter	Hodnota
Connection Type	MQTT client
Mote Field Name	devEUI
Payload Field Name	dataHex
Payload Encoding	Base16 encoding
Time Field Name	timeStamp
Time Mask *	yyyy-mm-ddThh:mi:ss.mss
Frame Type Field Name	
Frame Type Field Required Value	
MQTT User Name	(poda AppEUI)
MQTT Password	***
MQTT Topic Filter	app/(appEUI)/node/+/rxdata
MQTT Client ID	D2000kom
MQTT Clean Session Flag	YES

MQTT Publish Format	{"reference":"","confirmed":true,"fPort":3,"dataHex":#PAY#}
MQTT Publish QoS	QoS_0
MQTT Ping Interval	60
MQTT Reply Timeout	20
MQTT Wait Timeout	00.100
MQTT Max. Wait Retry	3

* Note.: Timestamp is sent in local time. Station time settings are to be configured accordingly.

Line parameters tested for Connection Type=*Sigfox Client (JSON via HTTPS)* towards Sigfox cloud

Parameter	Hodnota
Connection Type	Sigfox Client (JSON via HTTPS)
Mote Field Name	data[].device
Payload Field Name	data[].data
Payload Encoding	Base16 encoding (Slovanet, Loriot, Sigfox)
Time Field Name	data[].time
Time Mask	UNIX
Frame Type Field Name	
Frame Type Field Required Value	

Communication station configuration

- Communication protocol "**IoT over LoRaWAN/Sigfox**".
- Station address: the address of the station is the identifier of the specific device (mote) that is in the Mote Field Name field.
  - for Connection Type=*Kerlink IoT Station SPN* address is a text representation of an 8-byte LoRaWAN address (e.g. 00-00-00-00-21-1a-e3-c8)
  - for Connection Type=*MQTT Client* the address may be a text representation of an 8-byte LoRaWAN address (e.g. 0018B2000000147D) or a symbolic address defined within the MQTT server (e.g. *fieldtestdevice*)
  - for Connection Type=*Sigfox Client (JSON via HTTPS)* address is a device identifier (e.g. 21FDA5)

Station parameters

Dialog station configuration - **Protocol Parameter** field.
They affect some optional protocol parameters. The following station parameter parameters can be entered:

Table 2

Parameter	Description	Unit	Default Value
Device Type	Type of device. Each device type may have its own structure of transmitted data (payload). The list of supported devices will gradually increase. Currently supported devices are: • **None** - no device • **OEM device** - payload parsing is performed by an external library (dll) • **Adeunis RF Field Test Device** - test device sending GPS position data and temperature data • **SolidusTech IndoorUNI Sensor** - indoor temperature and humidity meter • **SolidusTech miniUNI DS18B20 Sensor** - temperature meter for outdoor use • **Adeunis RF LoRaWAN TEMP (ARF8180BA)** - temperature meter for outdoor use with two independent temperature sensors • **Codea WZU-SG (Landis+Gyr Ultraheat T550)** - radio module WZU-SG by Codea for heat meter Landis+Gyr UH50/UC50/T550 • **Moire Labs P1AP/P1AT devices** - temperature and pressure sensors from Moire Labs (https://iotransducer.com)	-	None
External Dll Name	Name of external DLL library with code for payload parsing for Device Type=*OEM device*.	-	

No Data Timeout	Timeout after which the station goes into a communication error state if no data has been received.	hh:mi:ss	01:00:00
MQTT Topic (for writing)	The topic used when writing the value (for Connection Type=*MQTT client*). **Note:** for *ipesoft-test* user and *fieldtestdevice* device writing was tested towards TheThings.network with MQTT_TOPIC=*ipesoft-test/devices /fieldtestdevice/down.*	-	

Sigfox Download Command File	For Sigfox: the path to the file for downloading data from the Sigfox cloud to the input directory (specified by parameter *Input file* of the *File I/O* line) together with possible parameters. The path may contain (like the *Input File* parameter) the symbolic constant **#APPDIR#** (application directory) as well as **#ADDR#** (station address) so that a single file can be used to handle multiple stations.
	Example for Windows: *#APPDIR#/Sigfox_cmd/get.bat #ADDR#*
	Example for Linux: */bin/sh #APPDIR#/Sigfox_cmd/get.sh #ADDR#*
	Note: this parameter does not have to be entered if an independent mechanism is used to download the data.
	
	The download itself can use the curl utility to perform an HTTPS GET query against the Sigfox web server.
	
	Example for Windows platform - file *get.bat* (download is via a proxy server, server identity verification is disabled, *xxx:yyy* is user name and password for Sigfox cloud):
	
	rem default count=1, possible to download up to 100 values
	set count=1
	rem ID of the device is 1st parameter
	set id=%1
	c:\utils\curl.exe --proxy http://proxy:3128 --insecure -u xxx:yyy -o msg_%id%.txt https://backend.sigfox.com/api/devices/%id%/messages?limit=%count%
	
	Example for Linux platform - file *get.sh* (it is necessary to explicitly specify that files are downloaded e.g. to *Sigfox_input* directory):
	
	#/bin/sh# default count=1, possible to download upto 100 values
	count=1
	base=$(dirname "$0")
	#ID of device is 1st parameter
	id=$1
	#name of downloaded data file (including directory)
	datafile=$base/../Sigfox_input/msg_$id.txt
	/usr/bin/curl --proxy http://proxy:3128 --insecure -u xxx:yyy -o $datafile https://backend.sigfox.com/api/devices/$id/messages?limit=$count
	
	Note: if command files are edited prior to running the D2000 KOM process, parameter *count* can be increased up to 100, resulting in downloading not only the latest value but also previous (up to 99) historical values. Then, after the first polling, the parameter *count* can be decreased to 1.		

Sigfox Download Timeout	For Sigfox: timeout for downloading data via Sigfox Download Command File. If downloading takes longer, the station will go to error.	sec	30

I/O tag configuration

Possible value types of I/O tags: **Ai**, **Di**, **Ci**, **TxtI**, **Qi**, **TxtO**.

Value type	Address (address type)	Description
Ai, Di, Ci, Qi, TxtI	Payload	I/O tags parsed from the payload. Address (Address) depends on the device type (Device Type parameter). The address is not case sensitive. A special case is a blank address - the I/O tag will contain the entire payload (after relevant decoding depending on the Payload Encoding parameter). The following tables indicate the addresses for each device type: Payload addresses for the device type of **OEM Device:** The address depends on the specific implementation (dll library). Payload addresses for device type **Adeunis RF Field Test Device**

Address	Description
Status	Status byte of device.
TriggerAccelerometer	The True value means that the sending of the message was initiated by an accelerometer.
TriggerButton	The True value means that the sending of the message was initiated by a button.
Temperature	Measured temperature (-128 °C .. 127°C).
GpsLatitude	Latitude (0-90 degrees) from the GPS sensor. **Note: GPS data may be missing if the device has no GPS signal.**
HemisphereSouth	The True value means that the latitude is south (the device is in the southern hemisphere).
GpsLongitude	Longitude (0-180 degrees) from the GPS sensor.
HemisphereWest	The True value means that the latitude is west (the device is in the western hemisphere).
GpsQualityReception	GPS signal reception quality: 1 Good, 2 Average, 3 Poor
QpsQualitySatellites	The number of visible GPS satellites.
UplinkCounter	Packet uplink counter (packets sent from the device to the LoRaWAN gateway).
DownlinkCounter	Packet uplink counter (packets sent to the device from the LoRaWAN gateway).
BatteryLevel	Battery voltage in mV.
RSSI	Indicator of the strength of the received signal (Received Signal Strength Indicator) - value between 0-255. The payload contains the field only if a write has been previously performed to the device (sending data from the LoRaWAN gateway to the device).
SNR	Signal Noise Ratio v dB (-128 .. 127). The payload contains the field only if a write has been previously performed to the device (sending data from the LoRaWAN gateway to the device).

Payload addresses for device type **SolidusTech IndoorUNI Sensor**

Address	Description
ADR	Adaptive Data Rate (optimizing data transfer speed and energy consumption). Value True indicates that ADR is on.
DataRate	Data Rate (data transmission rate) 0-5.
SNR	Signal Noise Ratio v dB (-128 .. 128).
BatteryLevel	Battery voltage in mV.
Temperature	Temperature (-125.99°C .. 125.99°C) with 0.1°C resolution.
Humidity	Relative humidity (0.0%-100%) with 0.1% resolution.
PowerAdapter	The True value means that the device is connected to a power adapter, the False value means that it is powered by a battery (always False for firmware version FW 0.2.2 and lower).
Contact	The True value means that an auxiliary contact is switched on (always False for firmware version FW 0.2.2 and lower).

Payload addresses for device type **SolidusTech miniUNI DS18B20 Sensor**

Address	Description
BatteryLevel	Battery voltage in mV.
SNR	Signal Noise Ratio of the previous payload in dB. It applies after ACK is received. Value 127 indicates an undefined value (no ACK or downlink packet was received from LoRaWAN gateway).
Temperature	Temperature (-25°C .. 85°C) with resolution 0.1°C.

Payload addresses for device type **Adeunis RF LoRaWAN TEMP (ARF8180BA)**

Address	Description
FrameCounter	Internal message counter going from 0 to 7.
BatteryLow	Low battery indicator. Has values True or False.
HWError	Indicator of hardware error in a device (temperature sensor error etc.).
InternalTemp	The value of the temperature sensor located in the device housing with a resolution of 0.1 °C.
ExternalTemp	The value of the temperature sensor located on an external wire with a resolution of 0.1 °C.

Payload addresses for device type **Codea WZU-SG (Landis+Gyr Ultraheat T550)**

Address	Description
Energy	Current amount of energy (in hundredths of GJ)
Volume	Current volume (in tenth of m3)
ErrorFlag	Error flag
MissingTime	Number of error hours (in hours)
Status	Status of module-reason for sending data: 0x00 - Ok 0x10 - Error of reading a meter (bat format of E,V data..) 0x20 - Install 0x30 - JMP 0x40 - Error of meter

Payload addresses for device type **Moire Labs P1AP/P1AT devices**

Address	Description
Unit	Unit (0-°C, 1-Pa, 2-kPa, 3-MPa)
MinValueRaw	Minimum temperature/pressure value (number from communication)
MaxValueRaw	Maximum temperature/pressure value (number from communication)
AvgValueRaw	Average temperature/pressure value (number from communication)
LastValueRaw	Last temperature/pressure value (number from communication)
MinValue	Minimum temperature/pressure value (in case of pressure a number converted to Pa)
MaxValue	Maximum temperature/pressure value (in case of pressure a number converted to Pa)
AvgValue	Average temperature/pressure value (in case of pressure a number converted to Pa)
LastValue	Last temperature/pressure value (in case of pressure a number converted to Pa)
Boot	The value is set after receiving the message that the meter sends after the start Note: in the *Filter* tab it is necessary to set *New time => new value*.
Battery	Battery charge level (0-255).

<table>
<tr>
<td>Ai, Di, Ci, Qi, TxtI</td>
<td>Envelope</td>
<td>I/O tag parsed from the envelope of the message. The address is the name of the field in the envelope of the message.
Note: For JSON messages that can be structured, the syntax level1.level2.level3 ... is supported, e.g. rx.moteeui and if they contain fields (indexed from 1) then also level1[index1].level2[index2].level3 ... syntax, e.g. rx.gwrx[1].time.

Example of JSON message for Connection Type=Kerlink IoT Station SPN (added spacing and alignment for better legibility):

<pre>{
  "rx": {
    "moteeui": "00-00-00-00-00-1e-fc-1d",
    "userdata": {
        "seqno": 77,
        "port": 1,
        "payload": "NzM3RjAwZTgwMA==",
        "motetx": {
            "freq": 868500000,
            "modu": "LoRa",
            "datr": "SF7BW125",
            "codr": "4/5"
        }
     },
     "gwrx": [
        {
         "time": "2017-07-05 16:06:52",
         "chan": 2,
         "rfch": 0,
         "rssi": -33,
         "lsnr": 7.5
        }
     ]
  }
}</pre>
I/O tags of envelope may have addresses e.g. rx.moteeui, rx.userdata.seqno, rx.userdata.motetx.freq, rx.gwrx[1].time.

Example of JSON message for Connection Type=MQTT Client (JSON via MQTT) (added spacing and alignment for better legibility):

<pre>{
  "app_id":"ipesoft-test",
  "dev_id":"fieldtestdevice",
  "hardware_serial":"0018B2000000147D",
  "port":2,
  "counter":549,
  "payload_raw":"niNJElVwAYQ5UBYfBBBN",
  "metadata":{
    "time":"2017-08-10T08:12:26.06860368Z",
    "frequency":867.5,
    "modulation":"LORA",
    "data_rate":"SF7BW125",
    "coding_rate":"4/5",
    "gateways":[
      {
        "gtw_id":"eui-000000000003080b",
        "timestamp":705621508,
        "time":"2017-08-10T08:12:26.434682Z",
        "channel":5,
        "rssi":-34,
        "snr":7.8,
        "latitude":49.20927,
        "longitude":18.73184,
        "altitude":359
      }
    ]
  }
}</pre>
I/O tags of envelope may have addresses e.g. dev_id, metadata.time, metadata.gateways[1].latitude.</td>
</tr>
<tr>
<td>TxtI</td>
<td>All data</td>
<td>I/O tag, that will contain the complete received message - the whole envelope (e.g. JSON message). The I/O tag is intended for debugging purposes and for eventual processing of the entire message in the script.</td>
</tr>
<tr>
<td>TxtO</td>
<td>Write (MQTT)</td>
<td>I/O tag for writing. Currently implemented just for Connection Type=MQTT client and tested towards cloud TheThings.Network. The value of the I/O tag is considered to be a payload that will be encoded (depending on the Payload Encoding parameter) and inserted into the message template defined by the MQTT Publish Format parameter, where it will replace the #PAY# string . The resulting message will be sent to the MQTT server.</td>
</tr>
</table>

Literature

Links
Official website of LoRaWAN alliance https://www.lora-alliance.org/technology
Official website of MQTT protocol http://mqtt.org

Specifications and Standards
MQTT 3.1.1 specification http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
ISO/IEC 20922:2016 http://www.iso.org/standard/69499.html

Descriptions of Data Formats and API
www.loriot.io - Application API Data Format https://www.loriot.io/home/documentation.html#docu/app-data-format
www.thethingsnetwork.org - API Reference https://www.thethingsnetwork.org/docs/applications/mqtt/api.html

Document revisions

- Ver. 1.0 - August 10th, 2017 - Document creation.
- Ver. 1.1 - August 25th, 2017 - Extended line configuration (Frame Type, Time Mask - UNIX, UNIXMS, PayloadEncoding - Base16), support of AdeunisRF LoRaWAN TEMP device and communication with Loriot.io.
- Ver. 1.2 - October 26th, 2018 - Added support for Sigfox protocol.

Related pages:

Communication protocols