**LoRaWAN Network Server Demonstration:**

**Inter-Server interface definition**

[1 History 3](#_Toc424030239)

[2 LoRa server interface diagram 4](#_Toc424030240)

[3 Interface list 5](#_Toc424030241)

[4 Transport protocols 5](#_Toc424030242)

[5 JSON objects 6](#_Toc424030243)

[6 General notes 6](#_Toc424030244)

[7 JSON object descriptions 6](#_Toc424030245)

[7.1 Key 6](#_Toc424030246)

[7.2 Tree diagram 7](#_Toc424030247)

[7.3 Table 8](#_Toc424030248)

[8 JSON object examples 14](#_Toc424030249)

[8.1 Join 14](#_Toc424030250)

[8.1.1 From Gateway to NS 14](#_Toc424030251)

[8.1.2 From NS to AS 14](#_Toc424030252)

[8.1.3 From AS to NS 15](#_Toc424030253)

[8.1.4 From NS to AS 15](#_Toc424030254)

[8.1.5 From AS to NS 15](#_Toc424030255)

[8.1.6 From NS to Gateway 16](#_Toc424030256)

[8.1.7 AS to CS 16](#_Toc424030257)

[8.2 Receive upstream data 17](#_Toc424030258)

[8.2.1 Gateway to NS 17](#_Toc424030259)

[8.2.2 NS to AS 17](#_Toc424030260)

[8.2.3 AS to CS 18](#_Toc424030261)

[8.3 Transmit downstream data 19](#_Toc424030262)

[8.3.1 CS to AS 19](#_Toc424030263)

[8.3.2 AS to NS 19](#_Toc424030264)

[8.3.3 NS to AS 20](#_Toc424030265)

[8.3.4 AS to NS 20](#_Toc424030266)

[8.3.5 NS to Gateway 20](#_Toc424030267)

[8.3.6 NS to AS 21](#_Toc424030268)

[8.3.7 AS to CS 21](#_Toc424030269)

[8.4 Adaptive data rate 21](#_Toc424030270)

[8.4.1 NS to NC 21](#_Toc424030271)

[8.4.2 NC to NS 22](#_Toc424030272)

[8.5 Command dialogue 22](#_Toc424030273)

[8.5.1 From console to server 22](#_Toc424030274)

[8.5.2 From server to console 23](#_Toc424030275)

[9 Glo*s*sary 24](#_Toc424030276)

[10 References 28](#_Toc424030277)

# History

|  |  |  |
| --- | --- | --- |
| Revision | Modification / Remarks / Motive | Author |
| 1.0 | Document created | DRo |

# LoRa server interface diagram



Figure 1: Diagram of LoRa server interfaces

# Interface list

| Participants | Transport protocols | JSON Objects transmitted (A to B) | JSON Objects transmitted (B to A) |
| --- | --- | --- | --- |
| Gateway (A)  Network server (B) | Gateway message protocol/ UDP | rxpk, stat, | txpk |
| Network server (A)  Application server (B) | TCP | app.userdata,  mote.resetdetected,  mote.msgsent,  mote.msgsentfail  mote.ackrx,  mote.qlen,  mote.seqnoreq,  join.request,  join.details | app.userdata,  mote.qlenquery,  mote.seqnogrant  join.accept  join.complete |
| Application server (A)  Customer server (B) | TCP | app.userdata  mote.resetdetected,  mote.msgsent,  mote.msgsentfail  mote.maccmdsent,  mote.ackrx,  mote.qlen,  mote.join | app.userdata,  mote.qlenquery, |
| Network server (A)  Network controller (B) | TCP | maccmd,  mote.maccmdsent,  app.maccmd.transmit.cancelled,  app.maccmd.transmit.queuelength | app.maccmd.transmit.cancel  app.maccmd.transmit.queuequery,  command |
| Any server (A)  Command Console (B) | UDP | command, ackreq, ack | command, ackreq, ack |
| Any server (A) to any other server (B) | TCP | ip.whichport | ip.publishedport |

Table 1: Objects used during server interaction

# Transport protocols

When JSON is transported over either the Gateway Message Protocol or directly over UDP, the JSON message shall occupy the entire payload. A trailing, zero valued, octet (Hex 0x00) is permitted but not required.

When JSON is transported over TCP, successive top level JSON objects shall be separated by a zero valued octet (Hex 0x00).

# JSON objects

JSON (JavaScript Object Notation) is a text based method of representing name, value pairs. The value of an object may itself be a JSON object.

When the JSON message contains a BASE 64 value, the transmitter shall not transmit padding characters. The receiver shall accept them, if present.

A hexadecimal value shall be transmitted using either case of the letters ‘a’ to ‘f’ to represent the hexadecimal digits greater than ‘9’. No separation characters (e.g. ‘:’ or ‘-‘) shall be transmitted.

# General notes

User data transmitted between the NS and the AS is encrypted.

All communication between the AS and the CS is unencrypted.

All numbers are transmitted in base 10, unless specified to be in base 16.

When a hexadecimal (base sixteen) value is transmitted it shall is transmitted as a text string. The letters 'a'-'f' (in either case) represent values '10' to '16' respectively. The characters "0x" may be present immediately before the first (most significant) digit but are not required.

Signed positive values may or may not be prefixed by '+'.

Negative values shall be prefixed by '-'.

# JSON object descriptions

## Key

[member] indicates an array of objects of type [member]. When the array contains only one element, it is not required that the array is transmitted.

*italic* characters indicate that an object is not required

## Tree diagram



Figure 2: JSON object tree diagram, showing principal relationships

## Table

| Parent | Name | Type/Contains | Notes |
| --- | --- | --- | --- |
| Top Level |  |  |  |
| top level | app | moteeui, userdata, token, dir, motetx, [gwrx] |  |
| top level | maccmd | moteeui, *command,* token |  |
| top level | mote | eui, app, resetdetected, msgsent, mote.msgsentfail, ackrx, qlenquery, qlen, seqnoreq, seqnogrant, ,join, posn | The "app" element must be transmitted if any of "msgsent", "mote.msgsentfail", "ackrx", "qlenquery", "qlen" are transmitted. |
| top level | ackreq | unsigned integer (<216) | Requests that an ‘ack’ JSON object, containing the same integer, be transmitted to the sender. This allows the sender to request acknowledgement of a JSON message. |
| top level | ack | unsigned integer (<216) | Sent in response to an ‘ackreq’ message |
| top level | command | string | The text of the command |
| top level | join | appeui, moteeui, request, accept, details, complete | 'appEui' and 'moteEui' identify the mote and its application.  'request' and 'details' are sent only from NS to AS.  'accept' and 'complete' are sent only from AS to NS.  Only one of 'request', 'accept', 'details', 'complete' should be present.  Neither 'appeui' nor 'moteeui' are required when only 'request' is present.  'appeui' is only required for objects sent from the NS to the AS. |
| top level | gw | eui, *posn, loraregion* | 'eui' identifies the gateway. |
| top level | ip | whichport, publishedport |  |
| top level | rxpk | n/a | Defined in [1] |
| top level | txpk | n/a | Defined in [1] |
| top level | stat | n/a | Defined in [1] |
| app or maccmd Objects |  |  |  |
| app OR maccmd | moteeui | string | The EUI of the mote, as a hexadecimal number. |
| app OR maccmd | token | unsigned integer (<216) | An arbitrary number generated by the CS, used to associate AS responses with CS commands. |
| app | dir | string | Either “dn” when the object is being sent toward the mote or "up" when the object is being sent away from the mote. |
| app | seqno | unsigned integer (<232) | The sequence number of the frame |
| app OR maccmd | userdata | port, payload, | The objects ‘port’ and ‘payload’ shall not be transmitted on the NS to Network Controller interface. |
| app | motetx | freq, datr, codr, adr | The radio characteristics of the mote’s transmission of the frame. |
| app | gwrx | eui, time, timefromgateway[[1]](#footnote-2), rssi, lsnr | The characteristics of the frame during its reception by the gateway |
| maccmd | command | string | The unencrypted content of the MAC command coded into Base 64 (defined by [2]) |
| app.userdata Objects |  |  |  |
| app.userdata | port | unsigned integer (<28) | The LoRa mote port number from which the frame was received, or from which the frame is to be transmitted. |
| app.userdata | payload | string | The payload of the frame coded into Base 64 (defined by [2]). Objects sent between the CS and AS are unencrypted; those sent between the AS and NS are encrypted. |
| app.userdata.motetx Objects |  |  |  |
| app.userdata.motetx | freq | Unsigned decimal real number (<10000), | The transmission frequency in units of MHz.  Contains at least one and no more than three digits before the decimal point and may contain up to six digits after the decimal point. If no digit follows the decimal point, the point may be omitted. ‘800’, ‘800.0’ and ‘800.000000’ are all valid values. |
| app.userdata.motetx | datr | string | If the modulation technique is LoRa, the string is of the form "SFnBWm", where ‘SF’ and ‘BW’ are literals and ‘n’ and ‘m’ are unsigned integers, ‘n’ represents the ‘spreading factor’ 7≤n≤12 and ‘m’ represents the modulation bandwidth in kHz, range m < 1000.  If the modulation technique is FSK, the string comprises an integer representing the data rate in bits per second. |
| app.userdata.motetx | codr | string | ECC code rate. "codr" comprises the string "k/n", where 'k' represents the carried bits and 'n' the total number of bits received, including those used by the error checking/correction algorithm. |
| app.userdata.motetx | adr | Boolean | True when ADR is enabled.  Used only in the ‘from mote’ direction. |
| app.userdata.gwrx Objects |  |  |  |
| app.userdata.gwrx | eui | string | The EUI of the receiving gateway, as a hexadecimal number. |
| app.userdata.gwrx | time | string | The receive time in time zone GMT/UTC. The format is YYYY-MM-DDTHH:MM:SS.sssssssss, where ‘YYYY’ represents the four digit year number, ‘MM’ represents the two month number (where ‘01’ represents January and ‘12’ represents December), ‘T’ is a literal value, ‘HH’ represents the two digit hour number, ‘SS.sssssssss’ represents the seconds time of reception, with a granularity of between 0 and 9 decimal places; if no digit follows the decimal point, the point may be omitted.  “2014-04-30T23:45:56.123456789” is an example value. |
| app.userdata.gwrx | timefromgateway | Boolean | True when the accompanying ‘time’ value is generated by the gateway |
| app.userdata.gwrx | rssi | signed integer (‑99≤rssi≤99) | Received signal strength, in units of dBm. |
| app.userdata.gwrx | lsnr | Signed decimal real number (<1000), | The signal to noise ratio, in units of dB.  Contains at least one and no more than two digits before the decimal point and either zero or one digit after the decimal point. If no digit follows the decimal point, the point may be omitted. ‘99.9’, ‘-99.9’, ‘+99.9’, ‘99.’ and ‘99’ are all valid values. |
| join Objects |  |  |  |
| join | moteeui | string | The EUI of the mote, as a hexadecimal number. |
| join | appeui | string | The EUI of the application, as a hexadecimal number. |
| join | request | frame | Sent from the NS to the AS. Contains the frame received from the mote |
| join | accept | Boolean | Sent from the AS to the NS. 'True' signifies that the AS accepts the mote. |
| join | details | moteaddr, devicenonce | Sent from the NS to the AS. Sends the data needed by the AS to generate the application and network session keys. |
| join | complete | frame, networkkey | Sent from the AS to the NS. Contains the ‘join accept’ frame that must be transmitted (unchanged) to the mote and the network session key, that the NS must use to authenticate frames received from the mote. |
| join.request OR join.complete | frame | string | The content of the LoRa frame, coded into Base 64 (defined by [2]). |
| join.details | moteaddr | string | The network address of the mote, as an 8 digit hexadecimal number. Leading zeros are optional. |
| join.details | devicenonce | integer < 216 | The nonce generated by the device. |
| join.complete | networkkey | string | The session key to be used by the NS to authenticate messages received from the mote to which it is allocated. The value comprises 16 hexadecimal digits. Leading zeros are optional. |
| gw Objects |  |  |  |
| gw | eui | string | The EUI of the gateway, as a hexadecimal number. |
| gw | posn | lati, longi, alti, tolh, tolv, gps | All the values are optional |
| gw | loraregion | string | The LoRa regional physical layer to which the gateway complies. One of americas902", "china779", "europe433", "europe863". |
| mote Objects |  |  |  |
| mote | eui | string | The EUI of the mote, as a hexadecimal number |
| mote | app | Boolean | True if the object refers to an application message; false if it refers to a MAC command. |
| mote | resetdetected | empty string | 'resetdetected' is sent by the NS when the server detects that the mote has been reset. |
| mote | msgsent | unsigned integer < 216 | Indicates that an application message or MAC command (corresponding to the token) has been transmitted to the mote |
| mote | msgsendfail | token, desc | The objects 'eui' and 'app' must precede 'msgsendfail'. |
| mote | ackrx | unsigned integer < 216 | Indicates that an acknowledgment message (corresponding to the application message indicated by the token) has been received from the mote |
| mote | qlenquery | empty string | Requests the number of messages waiting to be transmitted to the mote |
| mote | qlen | unsigned integer < 2 | The number of messages waiting to be transmitted to the mote |
| mote | seqnoreq | empty string | Sent from the NS to the AS requesting use of a sequence number, for the transmission of a frame that does not contain application data. |
| mote | seqnogrant | unsigned integer < 232 | Sent from the AS to the NS issuing a sequence number to be used for the transmission of a frame that does not contain application data. |
| mote | join | appeui | Sent from the AS to the CS, informing the CS that the AS has accepted a mote. |
| mote.join Objects |  |  |  |
| mote.join | appeui | string | The EUI of the application, as a hexadecimal number. |
| mote.msgsendfail Objects |  |  |  |
| mote.msgsendfail | token | unsigned integer < 216 | The token of the message whose transmission failed |
| mote.msgsendfail | desc | string | A description of the reason for the failure |
| posn Objects |  |  |  |
| posn | lati | double | The latitude of the position in units of degrees North of the equator |
| posn | longi | double | The longitude of the position in units of degrees East of the prime meridian. |
| posn | alti | double | The altitude of the position in units of metres above sea level (as defined by the United States' GPS system). |
| posn | tolh | double | The standard deviation of the horizontal tolerance of the position in metres. |
| posn | tolv | double | The standard deviation of the vertical tolerance of the position in meters |
| posn | loraregion | string | The LoRa regional physical layer to which the gateway complies. One of americas902", "china779", "europe433", "europe863". |
| ip Objects |  |  |  |
| ip | whichport | n/a | Requests the remote end of a TCP connection to return the ip.publishedport JSON object |
| ip | publishedport | unsigned integer < 216 | Reports the port number on which this server accepts connections |

Table 2: Table of JSON objects

# JSON object examples

The examples are indented for readability. The JSON objects shall be transmitted neither spaces nor non-printing characters.

## Join

### From Gateway to NS

{

"rxpk":

{

"tmst":20900514000,

"chan":2,

"rfch":0,

"freq":866.349812,

"stat":1,

"modu":"LORA",

"datr":"SF7BW125",

"codr":"4/6",

"rssi":-35,

"lsnr":5,

"size":23,

"data":"AMy7qgAAAAAATYMmmnj6AADl6YP1Jrw"

}

}

Received frame (docoded from Base64 encoded "data" )

================================

000 00 cc bb aa 00 00 00 00

008 00 4d 83 26 9a 78 fa 00

010 00 e5 e9 83 f5 26 bc

### From NS to AS

{

"join":

{

"request":

{

"frame":"AMy7qgAAAAAATYMmmnj6AADl6YP1Jrw"

}

}

}

### From AS to NS

{

"join":

{

"moteeui":"fa789a26834d",

"accept":true

}

}

### From NS to AS

{

"join":

{

"appeui":"aabbcc",

"moteeui":"fa789a26834d",

"details":

{

"moteaddr":"48000000",

"devicenonce":59877

}

}

}

### From AS to NS

{

"join":

{

"moteeui":"fa789a26834d",

"complete":

{

"frame":"ILmxdR1KHD/SVpqsg0FVFiw",

"networkkey":"e3e0d3a7b6cce87b158abe1b9316aeac"

}

}

}

### From NS to Gateway

{

"txpk":

{

"tmst":20902514000,

"freq":869.525000,

"rfch":0,

"powe":14,

"modu":"LORA",

"datr":"SF9BW125",

"codr":"4/5",

"ipol":true,

"size":17,

"data":"ILmxdR1KHD/SVpqsg0FVFiw"

}

}

Received frame (docoded from Base64 encoded "data" )

================================

000 20 b9 b1 75 1d 4a 1c 3f

008 d2 56 9a ac 83 41 55 16

010 2c

### AS to CS

{

"mote":

{

"eui":"fa789f000000",

"join":

{

"appeui":"aabbcc"

}

}

}

## Receive upstream data

### Gateway to NS

{

"rxpk":

{

"tmst":20900514000,

"chan":2,

"rfch":0,

"freq":866.349812,

"stat":1,

"modu":"LORA",

"datr":"SF7BW125",

"codr":"4/6",

"rssi":-35,

"lsnr":5,

"size":23,

"data":"AMy7qgAAAAAATYMmmnj6AADl6YP1Jrw"

}

}

Received frame (docoded from Base64 encoded "data" )

================================

000 00 cc bb aa 00 00 00 00

008 00 4d 83 26 9a 78 fa 00

010 00 e5 e9 83 f5 26 bc

### NS to AS

{

"app":

{

"moteeui":"fa789f000000",

"dir":"up",

"userdata":

{

"seqno":0,

"port":10,

"payload":"k5WH1t/8cqlur3JaWCoU7A9aUFI",

"motetx":

{

"freq":866.34,

"modu":"LORA",

"datr":"SF7BW125",

"codr":"4/6",

"adr":false

}

},

"gwrx":

[

{

"eui":"ed240b0000000000",

"time":"2014-10-20T13:18:48Z",

"timefromgateway":false,

"chan":2,

"rfch":0,

"rssi":-35,

"lsnr":5

},

{

"eui":"7f9eca0000000000",

"time":"2014-10-20T13:18:48Z",

"timefromgateway":false,

"chan":2,

"rfch":0,

"rssi":-35,

"lsnr":5

}

]

}

}

### AS to CS

{

"app":

{

"moteeui":"fa789f000000",

"dir":"up",

"userdata":

{

"seqno":0,

"port":10,

"payload":"d3d3LnNlbXRlY2guY29tAAAAAAM",

},

"motetx":

{

"freq":866.34,

"modu":"LORA",

"datr":"SF7BW125",

"codr":"4/6",

"adr":false

},

"gwrx":

[

{

"eui":"ed240b0000000000",

"time":"2014-10-20T13:18:48Z",

"timefromgateway":false,

"chan":2,

"rfch":0,

"rssi":-35,

"lsnr":5

},

{

"eui":"7f9eca0000000000",

"time":"2014-10-20T13:18:48Z",

"timefromgateway":false,

"chan":2,

"rfch":0,

"rssi":-35,

"lsnr":5

}

]

}

}

## Transmit downstream data

### CS to AS

{

"app":

{

"moteeui":"fa789f000000",

"token":56,

"userdata":

{

"dir":"dn",

"port":10,

"payload":"ESIz"

}

}

}

### AS to NS

{

"mote":

{

"eui":":"fa789f000000",

"seqnoreq":""

}

}

### NS to AS

{

"mote":

{

"eui":"fa789f000000",

"seqnogrant":34

}

}

### AS to NS

{

"app":

{

"moteeui":"fa789f000000",

"seqno":34,

"token":56,

"userdata":

{

"dir":"dn",

"seqno":34,

"port":10,

"payload":"C3m2"

}

}

}

### NS to Gateway

{

"txpk":

{

"tmst":21016645000,

"freq":869.525000,

"rfch":0,

"powe":14,

"modu":"LORA",

"datr":"SF9BW125",

"codr":"4/5",

"ipol":true,

"size":12,

"data":"QAAAAEggzc1wgJ8E"

}

}

### NS to AS

{"mote":

{

"eui":"fa789f000000",

"app":true,

"msgsent":56

}

}

{"mote":

{

"eui":"fa789f000000",

"app":true,

"ackrx":""

}

}

### AS to CS

{

"mote":

{

"eui":"fa789f000000",

"app":true,

"msgsent":56

}

}

{

"mote":

{

"eui":"fa789f000000",

"app":true,

"ackrx":""

}

}

## Adaptive data rate

### NS to NC

{

"app":

{

"moteeui":"fa789ad39295",

"dir":"up",

"seqno":9,

"motetx":

{

"freq":866.34,

"datr":"SF12BW125",

"codr":"4/6",

"adr":true

},

"gwrx":

[

{

"eui":"40120000000000",

"time":"2015-03-12T15:43:25Z",

"timefromgateway":false,

"chan":2,

"rfch":0,

"rssi":-10,

"lsnr":10

},

{

"eui":"140120000000000",

"time":"2015-03-12T15:43:25Z",

"timefromgateway":false,

"chan":2,

"rfch":0,

"rssi":-10,

"lsnr":10

}

]

}

}

### NC to NS

{

"maccmd":

{

"moteeui":"fa789ad39295",

"command":"A1EfAAA"

}

}

## Command dialogue

### From console to server

{

"ackreq":10,

"command":"ping"

}

### From server to console

{

"ack":10

}

{

"command":"Network server is alive (99.98.Dummy)"

}

# Glo*s*sary

ADR: Adaptive Data Rate. ADR observes the quality of the signal received by the mote and changes the mote's spreading factor and transmit power in order to optimise the time and energy required for the mote to transmit a frame.

Application: An application is identified by an 'application EUI'. Each mote is assigned to a single application. The remote server or servers to which information is forwarded (for example the AS to which an NS forwards are received frame) are configured for each application.

AS: The LoRa application server

ASCII: American Standard Code for Information Interchange. A widely used standard for representing Latin text, Arabic numerals and punctuation as binary values.

Base64: A method of encoding binary data into ASCII text. The LoRa system uses Base64 to transport LoRa frames in JSON objects. Base64 is defined by IETF RFC 4648 [2].

cB: centiBel. One tenth of the decibel defined by Bell Laboratories

cBm: centiBel relative to 1mW. A measure of power, relative to 1mW expressed in cB.

Class: A data structure in C++. A class is often used to represent a real world entity.

Command Console: The LoRa Command Console allows the LoRa servers to be configured.

Cryptographic hash: The generation of a hash code using a key which is known only to the sender and receiver or receivers. The transmission and recalculation of a cryptographic hash can be used to verify that the message content has not changed.

CS: The LoRa Customer Server

dB: decibel; a logarithmic ratio of power. Defined by Bell Laboratories

dBm A logarithmic measure of power, decibel relative to 1mW

Downstream: Toward the mote

End-device: Synonymous with 'mote'

EUI: Extended Unique Identifier. In this document 'EUI' refers to a value from the 'EUI-64' number space managed by the IEEE.

Gateway: A LoRa gateway is transmits LoRa frames to, and receives LoRa frames from, LoRa motes

GMT Greenwich Mean Time; also known as Co-ordinated Universal Time and Zulu

GNSS: Global Navigation Satellite System. The most well-known GNSS is GPS.

GPS: Global Positioning System. A Global Navigation Satellite System.

GWMP: Gateway message protocol. The protocol used the transport JSON objects between the network server and the gateways, defined by [1].

IEEE: Institution of Electrical and Electronic Engineers ([www.ieee.org](http://www.ieee.org)).

IETF: Internet Engineering Task Force ([www.ietf.org](http://www.ietf.org)).

IP: Internet Protocol

IP port address An IP address or host name and either a UDP or a TCP port number. This document represents a port address in the form <IP address>:<port number> or <host name>:<port number>. E.g. 1.2.3.4:4500 or a.com:4500.

Join: A colloquial name for 'Over the Air' activation.

Join request frame: A LoRa frame sent as the initial part of the OTA activation protocol. The frame contains the mote's EUI, its application's EUI and its device-nonce (a 16 bit random number).

Join accept frame A LoRa frame sent as the concluding part of the OTA activation protocol. The frame contains the mote's LoRa network address, its network Id and its application nonce (a 24 bit random number).

JSON: JavaScript Object Notation. JSON is a textual based method of representing name, value pairs. The value of an object may itself be a JSON object. Within LoRa, JSON objects contain only ASCII characters.

JSON object A JSON name, value pair.

Key: In cryptography, a key is a piece of information (a parameter) that determines the functional output of a cryptographic algorithm or cipher. Without a key, the algorithm would produce no useful result.

LoRa: Long Range. Defined by the LoRa Alliance

LoRa Alliance: The industry body that defines the LoRaWAN protocol. (<http://lora-alliance.org/>)

LoRa port: Any user data transmitted to or received from the mote is associated with a 'port' number. User data to or from LoRa Port 0 is MAC command or MAC status data. The remaining 255 LoRa port values are available to the mote user.

LoRaWAN: The protocol by which a LoRa mote will communicate with a LoRa gateway. LoRaWAN is defined by the LoRa Alliance.

MAC: Media Access Control

MAC command: A command transmitted to the mote. A MAC command is transmitted to the mote either in the LoRa frame 'header option' area or as user data to LoRa Port 0. Multiple commands may be transmitted in a single frame.

MAC status: Status information received from the mote. A MAC status message is transmitted by the mote either in the LoRa frame 'header option' area or as user data from LoRa Port 0. Multiple status messages may be transmitted in a single frame.

Metadata: LoRa Metadata refers to information about the transmission or reception of a LoRa frame.

Mote: A LoRa end device. A LoRa mote communicates with a LoRa Gateway using the LoRa MAC or LoRa WAN protocol.

Mutex: MUTual EXclusion: a software engineering construct that is 'grabbed' and 'released' by a thread. If a thread attempts to grab a mutex that has been grabbed but not released by another thread, the first mentioned thread will suspend until the second mentioned thread releases the mutex. This allows the programmer to ensure that certain sections of code (for example those than update or read data that is shared between threads) are fully executed by one thread before being entered by another.

MySQL: MySQL is an open source database engine available from <http://www.mysql.com/>

namespace: A construct within the C++ programing language, allowing the context of a name to be specified

NC: The LoRa network Controller

Network id: The 'network id' of a mote is its 'network address' shifted right by 25 bits, leaving 7 bit value.

Network address: The LoRa network address is a 32 bit value contained in the LoRa frame that identifies its source or destination mote. The network address need be unique only within the transmission range of a mote or gateway and is distinct from the mote EUI.

NS: The LoRa network Server

OTA: Over the air

Over the air: One of two methods of adding a LoRa mote to a LoRa network. In the OTA method, the mote is configured with a mote EUI, an application EUI and a 128 bit cypher key ('appKey'). Handshaking between the mote and the LoRa servers causes a 32 bit LoRa network address and two 128 bit session keys to be generated. One session key (the 'authentication' key) is known to the mote and the NS. The other (the 'encryption' key) is known to the mote and the AS.

Provisioning: A synonym for 'personalisation'

Process: A running computer program. A process cannot access the memory used by another. Processes are started and stopped independently of others.

Personalisation: One of two methods of adding a LoRa mote to a LoRa network. The mote is configured with its network address and its authentication and encryption keys. The mote's EUI is always equal to its network address and the application EUI is always zero.

RSSI: Received Signal Strength Indication. The power of the received signal, normally measured in dBm.

Rx: Receive

Semaphore: A software engineering construct. The semaphore is used within the LoRa servers to allow one thread to 'wait' (suspend) on the semaphore. When another thread 'posts' the semaphore, the semaphore wakes the thread (if any) that has been waiting longest. The semaphore mechanism allow implementation of queues, where a reading thread 'waits' and a writing thread can 'post'.

Suspend: A thread is suspended when it is not available to execute because it is waiting for an event to occur.

Signal quality: The signal quality is normally measured in dBm and is the sum of the SNR (measured in dB) and the RSSI (measured in dBm).

SNR: Ratio of signal power to noise power.

Spreading factor: A parameter of a LoRa transmission. Two to the power of 'spreading factor' 'on the air' bits are transmitted to represent each frame bit.

TCP: Transmission Control Protocol. A connection based protocol for transporting a sequence of bytes. While the connection exists, the content is guaranteed to be delivered in order and without loss or corruption.

Thread: An independent path of execution within a process. The threads of a process share access to memory within the process.

Transform: An element of a data flow diagram that transforms its inputs to generate one or more outputs (<http://en.wikipedia.org/wiki/Data_flow_diagram>)

Tx: Transmit

UDP: User Datagram protocol: a simple protocol for transporting data packets. Delivery is not guaranteed. In addition the order of receipt is not necessarily the same as the order of transmission.

Wake: A thread 'wake' reverses the action of 'suspending' a thread

upstream: Away from the mote

UTC Co-ordinated Universal Time; also known as Greenwich Mean Time and Zulu

# References

Each trademark is the property of its owner.

|  |  |
| --- | --- |
| [1] | Semtech Ltd, “LoRaWAN Network Server Demonstration: Gateway to Server Interface Definition,” 2015. |
| [2] | IETF, “The Base16, Base32, and Base64 Data Encodings,” October 2006. [Online]. Available: https://www.ietf.org/rfc/rfc4648.txt. |

© Semtech 2015

All rights reserved. Reproduction in whole or in part is prohibited without the prior written consent of the copyright owner. The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent or other industrial or intellectual property rights. Semtech assumes no responsibility or liability whatsoever for any failure or unexpected operation resulting from misuse, neglect improper installation, repair or improper handling or unusual physical or electrical stress including, but not limited to, exposure to parameters beyond the specified maximum ratings or operation outside the specified range.

SEMTECH PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF SEMTECH PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE UNDERTAKEN SOLELY AT THE CUSTOMER’S OWN RISK. Should a customer purchase or use Semtech products for any such unauthorized application, the customer shall indemnify and hold Semtech and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs damages and attorney fees which could arise.

|  |  |
| --- | --- |
|  | |
| **Contact Information** |  |
|  |
| Semtech Corporation  Wireless Sensing and Timing Products Division  200 Flynn Road, Camarillo, CA 93012  Phone: (805) 498-2111 Fax: (805) 498-3804  E-mail: [support\_rf\_na@semtech.com](mailto:support_rf_na@semtech.com)  Internet: <http://www.semtech.com> | |
|  | |

1. The JSON values ‘chan’ and ‘rfch’, which are present in the ‘rxpk’ JSON object received from the gateway are not omitted from this JSON object [↑](#footnote-ref-2)