**AN.LS.50**

**LoRa load tester description**

## Table of Contents

[0.1 Table of Contents 2](#_Toc414633123)

[1 Introduction 3](#_Toc414633124)

[2 Start parameters 3](#_Toc414633125)

[3 Command File commands 3](#_Toc414633126)

[4 Example command file 7](#_Toc414633127)

[5 Reception file data 8](#_Toc414633128)

[5.1 Units 8](#_Toc414633129)

[5.2 Example 8](#_Toc414633130)

[6 Glossary 8](#_Toc414633131)

[7 References 13](#_Toc414633132)

# Introduction

Each command in **bold** is followed by one or more examples in *italics*

<> indicates a parameter

[] indicates optional text

| indicates choices

EUIs, keys and network addresses must be entered in hex. Colon and dash separators are optional. E.g. 12:34, 1:2:34:567-89A0 and 1334abdf are all valid.

Masks must be entered in hexadecimal. They must be an unbroken series of 1’s. I.e. The leftmost character must be ‘1’, ‘3’, ‘7’ or ‘F’ and all other characters must be ‘F’.

# Start parameters

LoadGenerator [ns <network server] [<verbosity>] <command file>

The optional 'network server' parameter gives the address of the network server. It is in the format machine name or IP address, followed by a colon and the port number. If the port number and colon are missing the default is used. If this parameter is omitted, the 'set ns' command file command must be included before the load tester communicates with the network server.

The optional parameter 'verbosity' is one of –verbose –monitor –minor –major

The command file name is always the last parameter

Examples:

loracmd ns iot.dev.semnet.dom commandfile.txt

loracmd commandfile.txt

loracmd ns iot.dev.semnet.dom:1680 –verbose commandfile.txt

loracmd –verbose ns 192.168.0.20:1680 commandfile.txt

# Command File commands

# comment

Comment markers can be anywhere on the line

*# this is a comment*

set ns <address>

*Sets the node address of the network server, and (optionally) its port address. Any of the following formats is acceptable 1.2.3.4, a.b.com, 1.2.3.4:1700, a.b.com:1700.*

set defaultGatewayRegion americas902|china779|europe433|europe863

*Sets the region of the motes that will be created on the load tester and sets the default region of the gateways that will be created on the network server.*

window 0|1

*Sets the mote receive window used by the network server. Only values 0 and 1 are acceptable*

config ns <command>

Issues configuration commands to the network server.

config ns deleteAllData yesREALLY

config <address, including port> <command>

Issues configuration commands to either an addressed server (e.g. the Application Server, Customer Server or Network Controller.

config 127.0.0.1:4000 deleteAllData yesREALLY

gateway add <number of gateways> [<EUI of first gateway>]

Adds a number of gateways. If EUI of first gateway is specified, the EUIs are set contiguously from that value. If not, the EUIs are set randomly from 10:00:00:00 to 10:FF:FF:FF

gateway add 20 #adds 20 gateways – random EUIs

gateway add 20 12:34:00:00 #adds 20 gateways – EUIs from 12:34:00:00 to 12:34:00:13

gateway position <eui> lat <latitude> long <longitude> alt <altitude>

Sets the position of gateway <eui>. Co-ordinates are in degrees; North and East are positive. Altitude is in metres.

gateway position 12:40:01 lat 51.501191 long -0.142003 alt 15

app add <app eui> <app server address inc port> <cust server address inc port> [<network controller address inc port>] <app name> [<app owner>]

Adds an application to the network and application servers.

app add 12:34:45 127.0.0.1:4000 127.0.0.1:5000 testapp myOwner

lt netaddr <start> <range>

Sets the range of network addresses that the LT will allocate to its configured motes. <start> and <range> are hexadecimal. The LT will configure the LoRa network address all subsequently created motes in the range <start> to <start> + <range> - 1. The user is responsible for ensuring that the allocated addresses do not clash with those allocated by the NS to OTA motes.

mote add <number> <base eui> <eui mask> app (<app eui>|<app name>) start <start-up rate> period <mean transmit period> [fsk] gateways [<mean number of gateways per mote>]

Adds a number of 'over the air' motes. Each mote will have a EUI in the range <base eui> to <base eui> + <mask> -1. The load tester assigns a random appkey and configures it in the app server. It assigns the mote to <app eui> or <app name>. The join requests are issued randomly over  seconds. If the 'gateways' parameter is present, the mote will be visible to a constant but random number of gateways (mean value <mean number of gateways per mote>. If the 'gateways' parameter is not present, the 'mote gateway add' command must be executed **and the 'mote' is created disabled**..

When the join has occurred the mote will transmit confirmed data frames with a mean period of <mean transmit period>

mote add 20 56:78:9A:00:00:00 0xFFFF app testapp start 1.0 period 3.0 gateways 3 fsk

mote32 add <number> <base address> < address mask> app (<app eui>|<app name>) period <mean transmit period> [fsk] [gateways <mean number of gateways per mote>]

Adds a number of 'provisioned' motes. Each mote will have a network address and EUI in the range <base address> to < base address> + <mask>-1. The mote is visible to a constant but random number of gateways (mean value <mean number of gateways per mote. The mote will transmit confirmed data frames with a mean period of <mean transmit period>

mote32 add 20 56:78:9A:00:00:00 0xFFFF app testapp start 1.0 period 3.0 gateways

mote enable all on|off

Enables data transmission on all motes

mote enable <eui> on|off

Enables data transmission on the indicated mote

mote confirm <eui> on|off|adrack

Sets the mote to transmit confirmed frame (on), unconfirmed frame (off) or unconfirmed frames but with the 'ADRACKReq' bit of the frame set (adrack).

mote gateway add <mote eui> <gateway eui>

Assigns a mote to be received by a gateway

reception <reception file>

Instructs the load tester to read the reception information contained in <reception file>

reception ReceptionFile.txt

run [<seconds>]

Runs the load tester for a number of seconds. If <seconds> is omitted, the load tester runs for ever.

printstatus

Prints the status of the gateways and motes

time <name>

Prints the time that the command was executed, relative to the start of the simulation.

quit

Terminates execution of the current file. If the command is not present, execution will continue until the file end and then terminate. If the current file has been called (using an 'include' command) execution returns to the calling file.

# Example command file

config ns deleteAllData yesREALLY

config 127.0.0.1:4000 deleteAllData yesREALLY

config 127.0.0.1:5000 deleteAllData yesREALLY

config 127.0.0.1:6000 deleteAllData yesREALLY

window 0

#config 127.0.0.1:5000 fileoutput DefaultTestOutput

#config 127.0.0.1:5000 sqloutput off

#config ns log monitor

#config 127.0.0.1:4000 log monitor

#config 127.0.0.1:5000 log monitor

app add aa:bb:cc 127.0.0.1:4000 127.0.0.1:5000 127.0.0.1:6000 testApp testOwner

app add 0 127.0.0.1:4000 127.0.0.1:5000 127.0.0.1:6000 defaultApp

gateway add 10 124000

gateway position 124000 lat 51.500502 long -0.124471 alt 10 #Palace of Westminster

gateway position 124001 lat 51.501191 long -0.142003 alt 15 #Buckingham Palace

gateway position 124002 lat 51.508020 long -0.076026 alt 8 #Tower of London

gateway position 124003 lat 51.531940 long -0.106265 alt 25 #The Angel

mote32 add 1 40:00 1 period 3

mote32 gateway add 40:00 12:40:00

mote32 gateway add 40:00 12:40:01

mote32 gateway add 40:00 12:40:02

reception ReceptionFile.txt

mote enable 40:00 on

mote enable all on

mote32 add 1 30:00 1 period 3 gateways 3 fsk

mote add 20 FA:78:9A:00:00:00 0xFFFFFF app testApp start 5 period 10.0 gateways 3

run 20

config 127.0.0.1:5000 mote send 30:00 port 20 data 11 22 33 44 55

run 20

config 127.0.0.1:5000 mote send 30:00 port 20 data 11 22 33 44 55

time start

run 6000

time finish

printstatus

# Reception file data

The Reception file contains a list of delays and attenuation and losses suffered during transmission from motes to gateways. Each line represents one transmission. When the lines relating to transmissions between a mote-gateway pair are exhausted, the final line defines the characteristics of all subsequent transmissions.

<mote eui> <gateway eui> att <antenuation> snr <received signal to noise ratio> delay <mote to gateway delay>

<mote eui> <gateway eui> lost

## Units

|  |  |
| --- | --- |
| Value | Unit |
| <attenuation> | dB |
| of <received signal to noise ratio> | dB |
| of <mote to gateway delay> | nanoseconds |

## Example

4000 124000 delay 12300 att 12 snr 5

4000 124000 lost

4000 124000 att 20 delay 10000 snr 10

# Glossary

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 a member of a single application. The configuration rules that set the remote server to which information is forwarded (for example the AS to which an NS forwards are received frame) are set 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 [1].

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, 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.

Exception: A programing construct in C++ where a thread encounters a 'throw' statement. The thread ceases to execute its existing function and any calling functions until a 'catch' statement is encountered.

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 know 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 the Semtech document ' Basic communication protocol between LoRa gateway and server'.

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 object A JSON name, value pair

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

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. 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.

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

Provisioning: A synonym for 'personalisation'

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] | IETF, “The Base16, Base32, and Base64 Data Encodings,” October 2006. [Online]. Available: https://www.ietf.org/rfc/rfc4648.txt. |
| [2] | Semtech Ltd, “LoRa gateway to network server interface definition AN.LS.13,” 2015. |
| [3] | Semtech Ltd, “LoRa IoT Reference Network Software Solution Configuration Manual AN.LS.12,” 2015. |
| [4] | LoRa Alliance, “LoRaWAN Specification,” LoRa Alliance, 2015. |
| [5] | Semtech Ltd, “LoRa IoT Reference Network Software Solution Inter-server Interface Definition AN.LS.14,” 2015. |
| [6] | Semtech Ltd, “LoRa IoT Reference Network Software Solution Server Build, Installation and Getting Started Guide AN.LS.11,” 2015. |

© 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> | |
|  | |