
Noma Documentation

Release 0.1.1

LNCM

Oct 30, 2019

Contents:

1	Command-line Usage	3
2	API Modules	5
2.1	node	5
2.2	bitcoind	6
2.3	lnd	7
2.4	install	8
2.5	usb	9
2.6	rpcauth	11
3	Indices and tables	13
	Python Module Index	15
	Index	17

CLI utility and Python API to manage bitcoin lightning nodes.

CHAPTER 1

Command-line Usage

node:

```
noma (info|start|stop|restart|logs|check|status)
noma (temp|swap|ram)
noma (freq|memory|voltage) [<device>]
noma usb-setup
noma tunnel <port> <host>
noma (backup|restore|source|diff|devtools)
noma reinstall [--full]
```

bitcoind:

```
noma bitcoind (start|stop|info|fastsync|status|check)
noma bitcoind get <key>
noma bitcoind set <key> <value>
noma bitcoind logs [--tail]
```

lnd:

```
noma lnd (start|stop|info)
noma lnd logs [--tail]
noma lnd connect <address>
noma lnd (create|unlock|status|check)
noma lnd lncli [<command>...]
noma lnd get <key> [<section>] [<path>]
noma lnd set <key> <value> [<section>] [<path>]
noma lnd autounlock
noma lnd autoconnect [<path>]
noma lnd lndconnectapp <hostport>
noma lnd lndconnectstring <hostport>
```

noma:

```
noma (-h|--help)
noma --version
```

```
Options:
-h --help      Show this screen.
--version      Show version.
```


2.1 node

Node hardware and software management related functionality

`noma.node.check()`
check box filesystem structure

`noma.node.devtools()`
Install common development tools, nano, tmux, git, etc

`noma.node.do_diff()`
Diff current system configuration state with original git repository

`noma.node.freq(device=)`
Get chip clock (default: arm)
Device str arm, core, h264, isp, v3d, uart, pwm, emmc, pixel, vec, hdmi, dpi
Return str chip frequency

`noma.node.full_reinstall()`
Full reinstall replaces entire FAT contents (boot partition), and the ext4 data contents as if we installed from freshly burned SD card

`noma.node.get_ram()`
Return amount of RAM

`noma.node.get_source()`
Get latest noma source code or update

`noma.node.get_swap()`
Return amount of swap

`noma.node.install_git()`
Install git

`noma.node.is_running (node=)`
Check if container is running
Return bool container is running

`noma.node.logs (node=)`
Tail logs of node specified, defaults to lnd

`noma.node.memory (device=)`
Get memory allocation split between cpu and gpu
Parameters device (*str*) – arm, gpu
Return str memory allocated

`noma.node.reinstall ()`
Regenerate box.apkovl.tar.gz and mark SD as uninstalled
Leaves FAT partition alone. kernel, kernel modules, containers, etc on the boot partition remain the same
Since there is less to download this method is faster than `reinstall -full`

`noma.node.start ()`
Start default docker compose

`noma.node.stop (timeout=1, retries=5)`
Check and wait for clean shutdown of lnd

`noma.node.temp ()`
Get CPU temperature
Return str CPU temperature

`noma.node.tunnel (port, hostname)`
Keep the SSH tunnel open, no matter what

`noma.node.voltage (device=)`
Get chip voltage (default: core)
Parameters device – core, sdram_c, sdram_i, sdram_p
Return str voltage

2.2 bitcoind

bitcoind related functionality

`noma.bitcoind.check ()`
Check bitcoind filesystem structure

`noma.bitcoind.create ()`
Create bitcoind directory structure and config file

`noma.bitcoind.fastsync ()`
Download blocks and chainstate snapshot
Return bool success status

`noma.bitcoind.generate_rpcauth (username, password=)`
Generate bitcoind rpcauth string from username and optional password

`noma.bitcoind.get_kv (key, config_path)`
Parse key-value config files and print out values

Parameters

- **key** – left part of key value pair
- **config_path** – path to file

Returns value of key

`noma.bitcoind.set_kv(key, value, config_path)`
 Set key to value in path kv pairs are separated by “=”

Parameters

- **key** – key to set
- **value** – value to set
- **config_path** – config file path

Return str string written

`noma.bitcoind.set_prune(prune_target, config_path=)`
 Set bitcoind prune target, minimum 550

`noma.bitcoind.set_rpcauth(config_path)`
 Write new rpc auth to bitcoind and lnd config

`noma.bitcoind.start()`
 Start bitcoind docker compose container

`noma.bitcoind.stop()`
 Stop bitcoind docker compose container, if running

2.3 lnd

LND related functionality

`noma.lnd.autoconnect(list_path=)`
 Auto-connect to a list of nodes in lnd/autoconnect.txt

`noma.lnd.autounlock()`
 Auto-unlock lnd using password.txt, tls.cert

`noma.lnd.backup()`
 Export and backup latest channel.db from lnd via ssh

`noma.lnd.check()`
 Check lnd filesystem structure

`noma.lnd.check_wallet()`
 This will either import an existing seed (or our own generated one), or use LND to create one. It will also create a password either randomly or use an existing password provided)

Return str Status

`noma.lnd.connectstring(hostname='192.168.83.33:10009', macaroon-
 file=PosixPath('/media/noma/lnd/neutrino/data/chain/bitcoin/mainnet/admin.macaroon'),
 tlsfile=PosixPath('/media/noma/lnd/neutrino/tls.cert'))`
 Show lndconnect string for remote wallets such as Zap

`noma.lnd.create_wallet()`

1. Check if there's already a wallet. If there is, then exit.

2. Check for password.txt
3. If doesn't exist then check for whether we should save the password (SAVE_PASSWORD_CONTROL_FILE exists) or not
4. If password.txt exists import password in.
5. If password.txt doesn't exist and we don't save the password, create a password and save it in temporary path as defined in PASSWORD_FILE_PATH
6. Now start the wallet creation. Look for a seed defined in SEED_FILENAME, if not existing then generate a wallet based on the seed by LND.

```
noma.lnd.encodemacaroons (macaroonfile=PosixPath('/media/noma/lnd/neutrino/data/chain/bitcoin/mainnet/admin.macaroon')
                           tlsfile=PosixPath('/media/noma/lnd/neutrino/tls.cert'))
    base64url encode macaroon and TLS certificate
```

```
noma.lnd.get_kv (key, section="", config_path="")
    Parse key-value config files and print out values
```

Parameters

- **key** – left part of key value pair
- **config_path** – path to config file
- **section** – [section] of the kv pair

Returns value of key

```
noma.lnd.randompass (string_length=10)
    Generate random password
```

```
noma.lnd.savepeers ()
    Save list of peers to file on disk for reconnecting
```

```
noma.lnd.set_bitcoind (password, user="", lnd_config="")
    Add bitcoind rpc username and password to lnd
```

```
noma.lnd.set_kv (key, value, section="", config_path="")
    Parse key-value config files and write them out with a key-value change
```

Note: comments are lost!

Parameters

- **key** – left part of key value pair
- **value** – right part of key value pair
- **section** – optional name of section to set in
- **config_path** – path to file

Returns

```
noma.lnd.setup_tor (version="")
    Add tor hidden service to lnd
```

2.4 install

Node installation related functionality

```
noma.install.apk_update ()
    Update apk mirror repositories
```

```
noma.install.check_for_destruction (device, path)
    Check devices for destruction flag. If so, format with ext4
```

`noma.install.check_installed` (*installed*=`'/media/mmcblk0p1/installed'`)
Check if LNCM-Box is installed

`noma.install.check_to_fetch` (*file_path*, *url*)
Check and fetch if necessary

`noma.install.create_swap` ()
Create swap on volatile usb device

`noma.install.enable_swap` ()
Enable swap at boot

`noma.install.fallback_mount` (*partition*, *path*)
Attempt to mount partition at path using ext4 first and falling back to any

Return bool success

`noma.install.install_apk_deps` ()
Install misc dependencies

`noma.install.install_firmware` ()
Install raspberry-pi firmware

`noma.install.mnt_any` (*device*, *path*)
Mount device at path using any filesystem

`noma.install.mnt_ext4` (*device*, *path*)
Mount device at path using ext4

`noma.install.move_cache` (*cache_dir*=`'/media/mmcblk0p1/cache'`, *var_cache*=`'/var/cache/apk'`)
Let apk cache live on persistent volume

`noma.install.setup_fstab` (*device*, *mount*)
Add device to fstab

`noma.install.setup_nginx` ()
Setup nginx paths and config files

`noma.install.usb_setup` ()
Perform setup on three usb devices

2.5 usb

USB and SD device related functionality

`noma.usb.dev_size` (*device*)
Return device size in bytes

Parameters *device* (*string*, e.g. `"sda"`) – device

Returns device size in bytes

Return type int

`noma.usb.fs_size` (*fs_path*)
Return filesystem size in bytes

Parameters *fs_path* – path to mounted filesystem

Returns filesystem size in bytes

`noma.usb.get_uuid` (*device*)
get uuid of device

`noma.usb.is_mounted(device)`

Check if a device is already mounted

Parameters `device` (*string*, e.g. `"sda1"`) – device or device + partition number

Returns True/False if device is mounted or not

Return type bool

`noma.usb.largest_part_size()`

get partition size in bytes of largest partition

`noma.usb.largest_partition()`

get largest device and partition name

`noma.usb.medium_partition()`

get second largest device and partition name

`noma.usb.sd_device_table()`

list sd devices

`noma.usb.sd_devs()`

list sd devices

`noma.usb.sd_part_size(partition)`

Return SD partition size in bytes

Parameters `partition` (*string*, e.g. `"sda"`) – device

Returns partition size in bytes

Return type int

`noma.usb.sd_partition_table()`

list sd partition sizes

`noma.usb.sd_partitions()`

list sd partitions

`noma.usb.smallest_partition()`

get third largest device and partition name

`noma.usb.sort_partitions()`

sort partitions from smallest to largest

`noma.usb.usb_device_table()`

list usb devices

`noma.usb.usb_devs()`

list usb devices

`noma.usb.usb_part_size(partition)`

Return USB partition size in bytes

Parameters `partition` (*string*, e.g. `"sda"`) – device

Returns partition size in bytes

Return type int

`noma.usb.usb_partition_table()`

list usb partition sizes

`noma.usb.usb_partitions()`

list usb partitions

`noma.usb.usb_setup()`
start usb-setup with 3 devices

`noma.usb.uuid_table()`
list UUIDs of all block devices e.g. {'sdc1': 'd641d2b9-4fcd-4c83-9415-7ca4e7553a5d'}

Returns dictionary of device names and UUIDs

2.6 rpcauth

`noma.rpcauth.generate_password()`
Create 32 byte b64 password

`noma.rpcauth.generate_salt(size)`
Create size byte hex salt

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`
- `search`

n

- `noma.bitcoind`, 6
- `noma.install`, 8
- `noma.lnd`, 7
- `noma.node`, 5
- `noma.rpcauth`, 11
- `noma.usb`, 9

A

apk_update() (in module *noma.install*), 8
autoconnect() (in module *noma.lnd*), 7
autounlock() (in module *noma.lnd*), 7

B

backup() (in module *noma.lnd*), 7

C

check() (in module *noma.bitcoind*), 6
check() (in module *noma.lnd*), 7
check() (in module *noma.node*), 5
check_for_destruction() (in module *noma.install*), 8
check_installed() (in module *noma.install*), 8
check_to_fetch() (in module *noma.install*), 9
check_wallet() (in module *noma.lnd*), 7
connectstring() (in module *noma.lnd*), 7
create() (in module *noma.bitcoind*), 6
create_swap() (in module *noma.install*), 9
create_wallet() (in module *noma.lnd*), 7

D

dev_size() (in module *noma.usb*), 9
devtools() (in module *noma.node*), 5
do_diff() (in module *noma.node*), 5

E

enable_swap() (in module *noma.install*), 9
encodemacaroons() (in module *noma.lnd*), 8

F

fallback_mount() (in module *noma.install*), 9
fastsync() (in module *noma.bitcoind*), 6
freq() (in module *noma.node*), 5
fs_size() (in module *noma.usb*), 9
full_reinstall() (in module *noma.node*), 5

G

generate_password() (in module *noma.rpcauth*), 11
generate_rpcauth() (in module *noma.bitcoind*), 6
generate_salt() (in module *noma.rpcauth*), 11
get_kv() (in module *noma.bitcoind*), 6
get_kv() (in module *noma.lnd*), 8
get_ram() (in module *noma.node*), 5
get_source() (in module *noma.node*), 5
get_swap() (in module *noma.node*), 5
get_uuid() (in module *noma.usb*), 9

I

install_apk_deps() (in module *noma.install*), 9
install_firmware() (in module *noma.install*), 9
install_git() (in module *noma.node*), 5
is_mounted() (in module *noma.usb*), 9
is_running() (in module *noma.node*), 5

L

largest_part_size() (in module *noma.usb*), 10
largest_partition() (in module *noma.usb*), 10
logs() (in module *noma.node*), 6

M

medium_partition() (in module *noma.usb*), 10
memory() (in module *noma.node*), 6
mnt_any() (in module *noma.install*), 9
mnt_ext4() (in module *noma.install*), 9
move_cache() (in module *noma.install*), 9

N

noma.bitcoind (module), 6
noma.install (module), 8
noma.lnd (module), 7
noma.node (module), 5
noma.rpcauth (module), 11
noma.usb (module), 9

R

`randompass()` (in module *noma.lnd*), 8
`reinstall()` (in module *noma.node*), 6

S

`savepeers()` (in module *noma.lnd*), 8
`sd_device_table()` (in module *noma.usb*), 10
`sd_devs()` (in module *noma.usb*), 10
`sd_part_size()` (in module *noma.usb*), 10
`sd_partition_table()` (in module *noma.usb*), 10
`sd_partitions()` (in module *noma.usb*), 10
`set_bitcoind()` (in module *noma.lnd*), 8
`set_kv()` (in module *noma.bitcoind*), 7
`set_kv()` (in module *noma.lnd*), 8
`set_prune()` (in module *noma.bitcoind*), 7
`set_rpcauth()` (in module *noma.bitcoind*), 7
`setup_fstab()` (in module *noma.install*), 9
`setup_nginx()` (in module *noma.install*), 9
`setup_tor()` (in module *noma.lnd*), 8
`smallest_partition()` (in module *noma.usb*), 10
`sort_partitions()` (in module *noma.usb*), 10
`start()` (in module *noma.bitcoind*), 7
`start()` (in module *noma.node*), 6
`stop()` (in module *noma.bitcoind*), 7
`stop()` (in module *noma.node*), 6

T

`temp()` (in module *noma.node*), 6
`tunnel()` (in module *noma.node*), 6

U

`usb_device_table()` (in module *noma.usb*), 10
`usb_devs()` (in module *noma.usb*), 10
`usb_part_size()` (in module *noma.usb*), 10
`usb_partition_table()` (in module *noma.usb*), 10
`usb_partitions()` (in module *noma.usb*), 10
`usb_setup()` (in module *noma.install*), 9
`usb_setup()` (in module *noma.usb*), 10
`uuid_table()` (in module *noma.usb*), 11

V

`voltage()` (in module *noma.node*), 6