egOS

Version:
1.8.2

Date:
07.09.2025

Contents

1 Introduction

Welotec egOS is our Linux-based operating system for Edge Gateway Series Products. During the design phase, we focused in particular on scalable edge applications and maximum IT security. We guarantee the constant reliability and security of our system through regular updates and upgrades. It is optimised for cloud connectivity and container deployment. As an open platform, our egOS is designed for your customised solutions and enables software-driven innovation.

This manual provides information on how to configure Welotec Edge Gateways running egOS. Please visit our documentation homepage for hardware manuals for specific devices.

2 Getting started

Please read the Chapter “Security Considerations” carefully before first use!

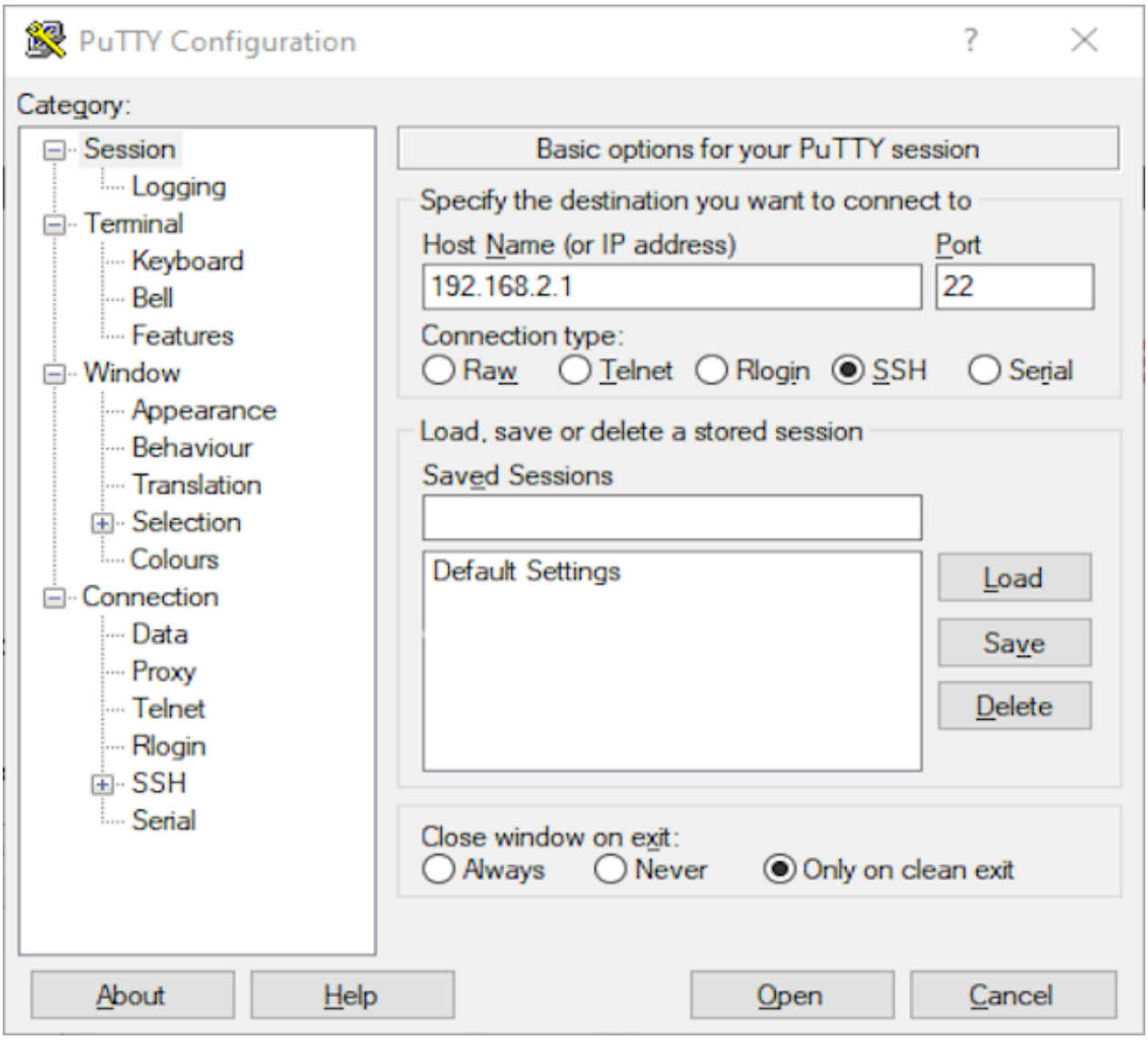
2.1 Connecting to the device

2.1.1 Power supply

Please check the hardware manual for your Edge Gateway model for power supply. Different hardware versions may use different voltages, connectors or pinouts. Connector for one hardware model, even if physically matching connector from other hardware model may be incompatible!

2.1.2 Access via ethernet using SSH

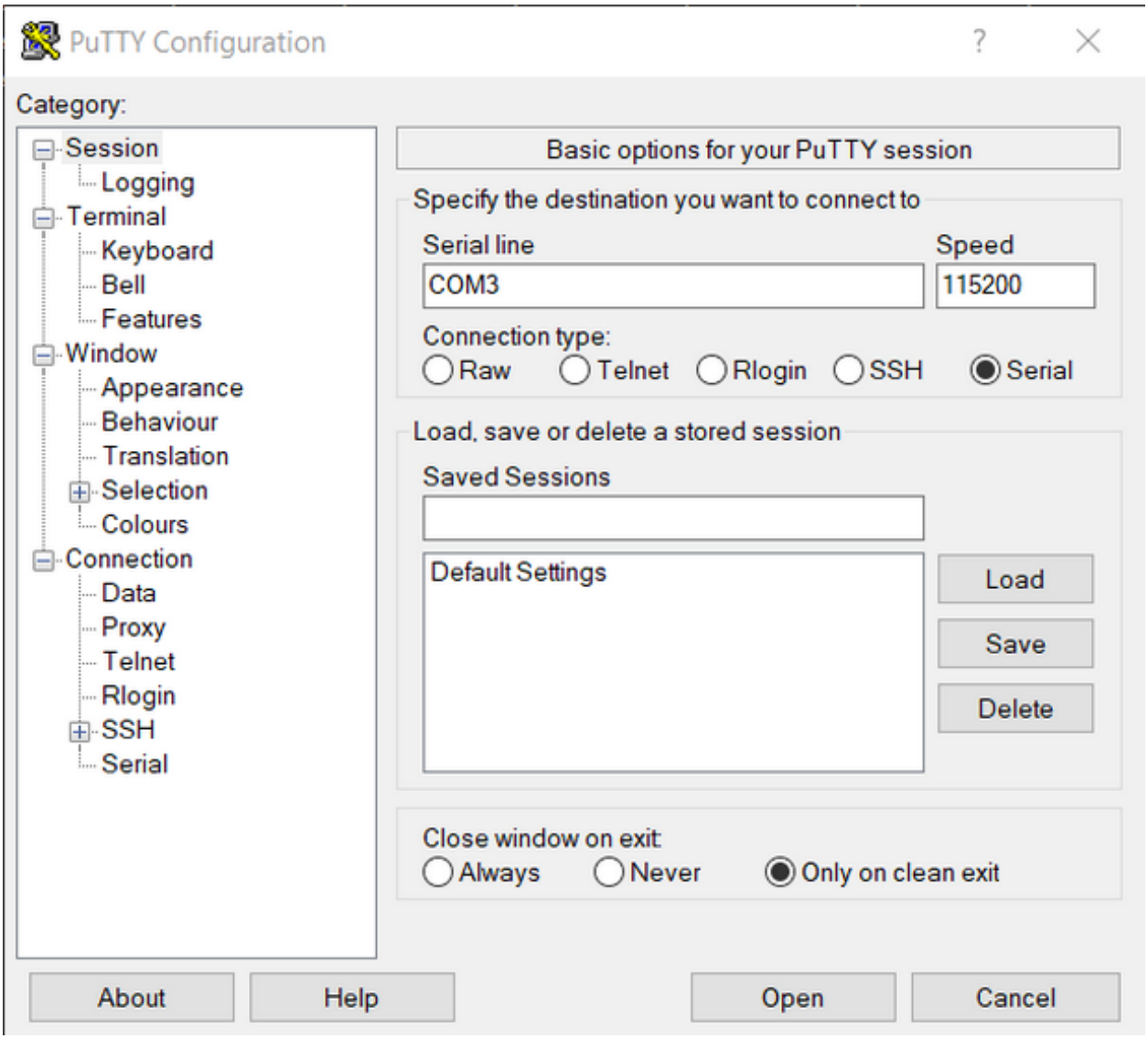
- Connect your computer to the Edge Gateway’s LAN1 port using an ethernet cable
- In factory default settings the Edge Gateway’s LAN1 is set to 192.168.2.1/24
- Configure your computer’s LAN port with an IP-configuration from 192.168.2.0/24 network (for example 192.168.2.2/24)
- Establish a connection using a terminal program (e.g. PuTTY) to 192.168.2.1 with port 22(SSH) Login with username “admin” and password “123456”
- Change password according to prompts



For security reasons it is crucial to perform first SSH connection using either direct cable connection between your computer and EG, or within secure network in which it impossible for the attacker to perform man in the middle attack. During first connection EG will advertise to SSH client (e.g. PuTTY) its public key. Users shall ensure that during any future connections exactly the same key is being shown to their SSH client, otherwise it is possible there is man in the middle attack in progress.

2.1.3 Access via COM-port using serial CLI

- In factory default settings the CLI is accessible via COM1 on the EdgeGateway
- Connect your computer's COM port to EdgeGateway's COM1 using a RS232 null modem cable
- Establish a connection using a terminal program (e.g. PuTTY) with your local COM-port and baud rate of 115200
- Login with username "admin" and password "123456"
- Change password according to prompts


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Serial line
Speed
COM3
115200
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Help
Open
Cancel

3 Security Considerations

This section provides detailed security recommendations to help you configure your Welotec IIoT Edge Gateway for optimal security in your IIoT deployment.

3.1 Exposed Interfaces and Services

In factory default the following interfaces and services are exposed and accessible:

Interface	Services Listening	Factory Default Setting	Comment
LAN1	SSH-Server	Static IP address: `192.168.2.1/24`	This is the only interface with preconfigured static IP address.
LAN2 … n	SSH-Server, DHCP-client	DHCP client ready to obtain IP configuration	This is the only interface that uses DHCP out-of-the-box.
Local Console	CLI	CLI	With attached keyboard and display
COM1	CLI	CLI redirected to COM port	/

Then LAN interface can be configured individually to a static IP-configuration or to obtain IP configuration from a DHCP server. Access Local Console as well as Console Redirect to COM-ports can be deactivated in configuration.

3.2 Device Security Features

3.2.1 Secure Boot and Encrypted Storage

Edge Gateways provide Secure Boot and the system and user data storage is fully encrypted with help of the TPM of the device. Boot partition is plain text and all of the boot data is secured against tampering with strong cryptographic signatures. Check the release notes about models and versions supporting this feature.

3.2.2 Firewall

The firewall of the device allows to limit the communication of the device to the necessary minimum for your use case. Please refer to *Firewall* Section for further details.

3.2.3 Security updates and patch management

Welotec is providing updates for the egOS regularly. Please refer to *OS Updates* Section for futher details.

3.3 Security Recommendations

3.3.1 Passwords

Strong passwords are the first line of defense against unauthorized access. You can disable Password based access to the device and only use SSH-host key authentication. If you want to use password based access it is reccomended to:

- Change the factory default password on first login
- Use passwords with a minimum length of 12 characters or more
- Use a combination of uppercase and lowercase letters, numbers, and special characters (e.g., !@#$%^&*)
- Do not use easily guessable patterns, such as sequences (e.g., “123456”, “abcdef”), repeated characters (e.g., “aaaaaa”), or dictionary words

3.3.2 Network Segmentation

Network segmentation is a critical security practice that involves dividing a network into smaller, isolated subnets or zones. This approach limits the impact of a security breach by preventing an attacker from moving laterally through the network and accessing critical systems. In an IIoT environment, this is crucial for protecting sensitive industrial control systems (ICS) and other operational technology (OT) assets. Use the Welotec IIoT Edge Gateway’s networking capabilities to create separate network segments. Methods for implementing segmentation:

- VLANs (Virtual LANs): Create VLANs to segment network traffic at Layer 2. This allows you to isolate devices on the same physical network.
- Subnets: Use IP subnets to divide the network at Layer 3. This provides logical separation and allows for different routing and firewall policies.
- Firewall Rules: Configure the gateway’s firewall to control traffic flow between different segments. Implement strict rules to allow only necessary communication and block all other traffic.
- Routing: Use static routes or dynamic routing protocols to control how traffic is routed between segments. Ensure that routing is configured to enforce security policies.

3.3.3 Secure remote access

Welotec is providing a software solution to enable Secure Remote Access: VPN Security Suite Please visit our homepage for further information.

3.3.4 Physical security of the device

- Place the device in a locked cabinet or implement other physical security measures to avoid manipulation of the device
- Limit the access to the device by disabling local login using the device ‘set_local_console’-command

3.4 Vulnerability Handling

Welotec has implemented a Coordinated Vulnerability Disclosure Policy - please visit the following site for further details: https://welotec.com/pages/coordinated-vulnerability-disclosure-policy

3.5 Secure Diposal

To securly dispose the device please reset it to factory defaults using the options provided here: *Factory reset*. This will delete all configuration, containers and user data on the device.

4 Configuration Management

The Edge Gateway provides the following options for device configuration:

- A Command Line Interface (CLI) including exporting and importing of full device configuration JSON
- Web Interface (disabled in factory default settings) for configuring device connectivity and supporting import of a full device configuration JSON
- Central device management solution SMART EMS - please visit our homepage for further information.

4.1 Command line interface

Following general commands are available:

```
device              generic configuration (not put into any of the specific commands or affecting␣
↪multiple of the specific domains at the same time)
docker              container management
docker-config       global docker configuration related actions (like exporting config of all␣
↪containers in EG)
nm                  network management in general
fw                  firewall configuration
ovpn                vpn tunnels management
```

Each of the commands has multiple subcommands and parameters described in this document.

4.1.1 Runtime help

Each command provides -h option which can be used at both: command level and subcommand level for quick refresh of what commands do and what parameters they take. Note that CLI command descriptions in this document are based mostly on runtime help system (so whenever you execute a CLI command with option –help the output will be identical to part of this document).

4.1.2 Autocompletion

For ease of working with parameters commands support bash autocompletion. To trigger autocompletion in bash use ‘Tab’ key. Note that bash autocompletion sometimes falls back to the file names even if file name is not really expected by the command syntax.

4.1.3 Argument globbing

Because of daemon based approach in some cases it is not trivial to provide autocompletion of dynamically changable values (like names of firewall rules), but globbing may be often used instead — when documentation of command says that globbing is allowed, patterns may be used instead of exact values. An asterisk (*) matches any string (including no characters at all), a question mark (?) matches any single character. Note that this is identical with shell globbing and shell may match a file name if a pattern is used. To prevent it, whole argument needs to be quoted or just special characters need to be prepended with backslash (\).

Compare below sequence of commands:

```
admin@eg ~> ls
admin@eg ~> fw preset list di*    # no files matching so di* is forwarded to daemon
{
    "saved": [
        "disabled"
    ],
    "being_edited": []
}
admin@eg ~> touch di.txt
admin@eg ~> fw preset list di*    # di* matches di.txt so daemon sees di.txt
{
    "saved": [],
    "being_edited": []
}
admin@eg ~> fw preset list 'di*'  # prevent file matching manually, daemon sees di*
{
    "saved": [
        "disabled"
    ],
    "being_edited": []
}
admin@eg ~>
```

4.2 Presets

Some of the commands deal with complicated sets of rules which work as intended only if they properly cooperate — good example of such area are firewall rules. Composing working firewall step by step, each applied immediately can easily lead to erratic and invalid behaviour of firewall. Additionally, potentially more than one person can have access rights allowing for management of EG and there needs to be a way to prevent one of them doing changes incompatible with other person's changes accidentally. For such areas preset system is provided, where each preset is independent working config, but only one of the presets can be applied to the whole device at the same time. Moreover, preset can be modified only when marked as being edited, which prevents accidental applying of non-finished preset.

4.2.1 Editing presets

- Subcommands "preset_edit" and "preset_create" leave preset in state of being edited
- Only preset being edited can be modified
- There are some factory prepared presets which cannot be put into edition state
- If there is more than one preset being edited at the same time then modification commands need to be given additional optional argument with name of preset intended for modification (usually '-n').

4.3 Confirmation of connectivity

EG is intended to be managed remotely, but some of the configuration commands can easily break connectivity to the device. Such commands after being applied require additional verification if the person who executed them still can access the device. For CLI this verification is done by displaying message — if connectivity is not lost, user will see it and will be able to respond. If there is no confirmation, changes will be rolled back. Unfortunately, in some cases rollback may not be ideal — there are multiple subsystems, so user is encouraged to always verify that configuration is as expected after such automatic rollback.

4.4 Exporting and importing configuration

The whole device configuration (or specific parts) can be exported to a JSON-File.

This file can be imported into other devices or deployed via SMART EMS — see get_config and set_config subcommands of various commands.

5 CLI Commands

5.1 Network management

This command allows to configure and check network related settings except firewall and ovpn.

This command has following subcommands:

```
    show                show network configuration
    status              show current network status
    cellular_turn_on    turn on cellular modem
    cellular_turn_off   turn off cellular modem
    cellular_configure  add configuration on the device for cellular modem
    wifi                manage wifi1 interface
    static_ip           set static ip for network interface
    dhcp                set dhcp on network interface
    ignore_default_route
                        enable/disable deafult route from DHCP
    set_config          set network configuration from a json file
    cellular_checklist  Run checks to fix cellular configuration
    get_config          save network configuration in ./network_config.json
    static_routing      Configure static routing
    promiscous_mode     enable/disable promiscous mode on a given interface
    ids                 enable/disable IDS on a given interface
```

Examples:

To configure cellular1 interface

```
  nm cellular_configure 1 --apn internet --pin 1234
```

To turn on ceullar1

```
  nm cellular_turn_on 1
```

To turn off ceullar1

```
  nm cellular_turn_off 1
```

Other examples:

```
nm static_ip --name lan1 --ip 10.0.0.2 --gateway 10.0.0.1 --subnet 24 --dns 9.8.8.8
nm static_ip --name lan1 --ip 10.0.0.2 --gateway 10.0.0.1 --subnet 24 --dns 1.1.1.1,2.2.2.2
nm static_ip --name lan1 --ip 10.0.0.2 --subnet 24
nm dhcp --name lan1
```

5.1.1 show

Print network related configurable values of EG to console. In case of configuration which may result in dynamically changing values it also shows them at the moment when command is executed. Commands `show`, `get_config` and `status` are related to checking network configuration. The difference is on their focus. `show` focuses on showing currently configured values to human. `get_config` focuses on whole configuration backup. `status` focues on current actual values used by system and includes also things that are not configurable (like loopback interface).

Synopsis:

```
nm show [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

the only way to call this command

```
  nm show
```

5.1.2 status

Same as executing classic Linux commands 'ifconfig' and 'route'. Commands `show`, `get_config` and `status` are related to checking network configuration. The difference is on their focus. `show` focuses on showing currently configured values to human. `get_config` focuses on whole configuration backup. `status` focues on current actual values used by system and includes also things that are not configurable (like loopback interface).

Synopsis:

```
nm status [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.1.3 cellular_turn_on

Enable connection on mobile network interface. This connection should be configured before turning on with help of `cellular_configure` command.

Synopsis:

```
nm cellular_turn_on [-h] interface
```

Detailed description of positional arguments:

```
  interface   Cellular interface number
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

to enable cellular1 interface

```
nm cellular_turn_on 1
```

to enable cellular2 interface

```
nm cellular_turn_on 2
```

5.1.4 cellular_turn_off

Disconnect mobile network interface (connected because of earlier `cellular_turn_on` command). Any active network connections going through this interface will be dropped.

Synopsis:

```
nm cellular_turn_off [-h] interface
```

Detailed description of positional arguments:

```
interface   Cellular interface name
```

Detailed description of named arguments:

```
-h, --help  show this help message and exit
```

Examples:

to disable cellular1 interface

```
nm cellular_turn_off 1
```

to disable cellular2 interface

```
nm cellular_turn_off 2
```

5.1.5 cellular_configure

Configures cellular connection parameters. Any previous configuration, will be removed, hence all of the parameters with non-default values should be provided each time this command is executed (otherwise default values will replace previously configured non-default ones). After configuration is done it can be checked with `show`, or exported with `get_config`.

Synopsis:

```
nm cellular_configure [-h] -a APN [-p PIN] [-A ACCESS_NUMBER] [-u USER]
                           [-P PASSWORD]
                           interface
```

Detailed description of positional arguments:

```
interface             Cellular interface number (starting from 1)
```

Detailed description of named arguments:

```
-h, --help            show this help message and exit
-a APN, --apn APN     APN name
-p PIN, --pin PIN     SIM card PIN, if not given means no PIN is configured
                      on SIM card. PIN can only contains digits. To remove
                      the PIN from the configuration, omit the "--pin"
                      argument.
```

(continues on next page)

(continued from previous page)

```
-A ACCESS_NUMBER, --access_number ACCESS_NUMBER
                      phone number of APN, if not given defults to *99***1#
-u USER, --user USER  user name for APN (if not given empty user name will
                      be used)
-P PASSWORD, --password PASSWORD
                      password for APN (if not given empty password will be
                      used)
```

Examples:

configure basic authentication using APN named ‘internet’ without PIN nor user/password authentication for cellular1 interface.

```
nm cellular_configure 1 --apn internet
```

configure authentication based on username ‘USER’ and password ‘PASSWORD’ with apn named ‘welo.vzwent’ but without PIN for SIM card for cellular2 interface

```
nm cellular_configure 2 --user USER --password PASSWORD --apn welo.vzwent
```

5.1.6 wifi

Manages wifi1 interface. Currently supported implementation is WiFi-Client role, which allows user to configure new connection, enable or disable it and scan for available networks.

This command has following subcommands:

```
   client    allows to use wifi1 interface in client role
```

Detailed description of named arguments:

```
-h, --help  show this help message and exit
```

Examples:

configure and connect connection profile with AP based on ssid ‘SSID’, key ‘KEY’ with authentication ‘wpa3-sae’ for wifi1 interface

```
nm wifi client config --ssid SSID --key KEY --authentication wpa3-sae
```

enable and connect connection profile on wifi1 interface

```
nm wifi client enable
```

disable and disconnect connection profile on wifi1 interface

```
nm wifi client disable
```

scan for available wifi networks

```
nm wifi client scan
```

client

This command has following subcommands:

```
   scan                 scan for available access points
   enable               enable and connect configured profile
   disable              disable and disconnect configured profile
   config               add and activate a new connection profile using the
                        given details
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
```

scan

Synopsis:

```
nm wifi client scan [-h] [--rescan {auto,yes,no}]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  --rescan {auto,yes,no}
                        used to either force or disable the scan regardless of
                        how fresh the access point list is
```

enable

Synopsis:

```
nm wifi client enable [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

disable

Synopsis:

```
nm wifi client disable [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

config

Synopsis:

```
nm wifi client config [-h] -s SSID -k KEY -a
                           {wpa-psk,wpa2-psk,wpa3-sae}
                           [-e {auto,ccmp,tkip} [{auto,ccmp,tkip} ...]]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -s SSID, --ssid SSID  ssid to connect to
  -k KEY, --key KEY     wpaX-key for the connection
  -a {wpa-psk,wpa2-psk,wpa3-sae}, --authentication {wpa-psk,wpa2-psk,wpa3-sae}
                        authentication method WPA-PSK / WPA2-PSK / WPA3-SAE
  -e {auto,ccmp,tkip} [{auto,ccmp,tkip} ...], --encryption {auto,ccmp,tkip} [{auto,ccmp,tkip} ...]
                        encryption Mode CCMP and/or TKIP for WPA / WPA2
```

5.1.7 static_ip

Allows to configure static IP addresses on given network interface. Replaces any previous configuration of interface (for example to remove gateway execute this command with same –ip parameter as currently set, but without –gateway parameter)

Synopsis:

```
nm static_ip [-h] --name {} --ip IP [--mtu MTU] [--gateway GATEWAY]
                  --subnet SUBNET [--dns DNS]
```

Detailed description of named arguments:

```
  -h, --help          show this help message and exit
  --name {}           network interface name
  --ip IP             ip address
  --mtu MTU           MTU
  --gateway GATEWAY   gateway (if not given, gateway will be set to null)
  --subnet SUBNET     subnet mask (number of bits, so use 24 instead of
                      255.255.255.0)
  --dns DNS           DNS to be used when interface is up (if not given, DNS
                      will not be associated with this interface being up and
                      set to null). You can also add multiple DNS servers at
                      once (comma separated).
```

Examples:

configure static ip 10.0.0.1/8 on lan1 without gateway!

```
nm static_ip --name lan1 --ip 10.0.0.1 --subnet 8
```

configure static ip 192.168.1.2/24 with gateway 192.168.1.1

```
nm static_ip --name lan2 --ip 192.168.1.2 --gateway 192.168.1.1
```

5.1.8 dhcp

Causes given network interface to use DHCP for obtaning network configuration. Any static configuration will be forgotten. First DHCP request will be sent immediately. Command will wait up to 8 seconds for connection to be up. No error reporting is performed — always verify connection status if you want to be sure DHCP worked as expected (e.g. by executing `nm {SHOW}`)

Synopsis:

```
nm dhcp [-h] --name {}
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
  --name {}   device name
```

Examples:

use DHCP for lan2

```
  nm dhcp --name lan2
```

5.1.9 ignore_default_route

Allows to ignore default route provided by DHCP (this may be required in case another default route is preferred).

Synopsis:

```
nm ignore_default_route [-h] -n NAME -i IGNORE
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  device name
  -i IGNORE, --ignore IGNORE
                        Yes to ignore route from DHCP
```

Examples:

default route provided by DHCP on lan2 will be ignored

```
  nm ignore_default_route --name lan2 -i yes
```

default route provided by DHCP on lan1 will be added to routing table

```
  nm ignore_default_route -n lan1 --ignore no
```

5.1.10 set_config

Replaces whole network configuration of EG with the one provided by json file. The file could have been generated on same or another device.

Synopsis:

```
nm set_config [-h] --filename FILENAME [--unconditionally]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  --filename FILENAME   json filename
  --unconditionally     Prevent connectivity checking after executing this
                        command (without this option user may be asked to
                        confirm that after command execution he has not lost
                        connection to the device, without such confirmation
                        command would be rolled back.)
```

Examples:

restore configuration from file ‘network_config.json’

```
  nm set_config --filename network_config.json
```

Example config file:

```
{
    "network": {
        "cellular1": {
            "access_number": "*99***1#",
            "apn": "",
            "password": "",
            "pin": "",
            "state": "off",
            "username": ""
        },
        "lan2": {
            "dhcp": true,
            "ignore_default_route": false
        },
        "lan1": {
            "dhcp": false,
            "ip": [
                "192.168.2.1"
            ],
            "subnet": [
                "24"
            ],
            "gateway": null,
            "dns": null
        }
    },
    "static_routing": {
        "enabled": false,
        "selected": "disabled",
        "saved": {
            "disabled": {}
        },
        "edited": {}
    }
}
```

5.1.11 cellular_checklist

Gather cellular state information

Synopsis:

```
nm cellular_checklist [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.1.12 get_config

Stores whole network related config for e.g. backup purposes into file ./network_config.json. Commands `show`, `get_config` and `status` are related to checking network configuration. The difference is on their focus. `show` focuses on showing currently configured values to human. `get_config` focuses on whole configuration backup. `status` focues on current actual values used by system and includes also things that are not configurable (like loopback interface).

Synopsis:

```
nm get_config [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

the only way to call this command

```
  nm get_config
```

5.1.13 static_routing

Manages static routing rules, both global and interface specific. Because routing rules can become quite complicated and to allow quick changes of configuration routing rules are stored in presets.

This command has following subcommands:

```
   print               print contents of preset being edited (whole or just a
                       part)
   add                 add new routing rule to edited preset
   remove              remove rule from global or interface config in edited
                       preset
   order               Change order of two elements
   enable              Enable routing rules in selected preset
   disable             Disable routing rules added by selected preset
   preset_create       create new preset in being_edited state
   preset_delete       delete preset currently being_edited
   preset_edit         mark saved preset as being_edited (allows to change
                       its contents, but prevents selecting it as current
                       one)
   preset_save         mark preset being_edited as saved
   preset_select       select saved preset for use
   preset_list         list existing presets (both saved and being_edited)
   preset_print        print contents of preset(s) (whole or just a part)
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
```

Examples:

enables static routing rules stored in currently selected preset

```
  nm static_routing enable
```

add route to 192.168.0.0/24 via `lan1` device

```
  nm static_routing add -n 192.168.0.0 -s 24 -d lan1
```

change order of rules for `lan1` device in edited preset

```
  nm static_routing order -i 1 -I 2 -c lan1
```

remove rule from `lan1` device in edited preset

```
  nm static_routing remove -i 1 -c lan1
```

print

Similar to `preset_print` but intended to print only single currently edited preset. If there is only one preset being edited name of the preset can be skipped. If there is more than one preset being edited, name of the preset must be given and it shall match only one of them. –part argument allows to limit output, e.g. in case of huge preset.

Synopsis:

```
nm static_routing print [-h] [-n NAME] [-p PART]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -p PART, --part PART  Part (at least container of rules) of preset which
                        shall be printed. Whole preset will printed if not
                        given.
```

Examples:

print contents of the one and only currently being edited preset

```
  nm static_routing print
```

add

Synopsis:

```
nm static_routing add [-h] [-m METRIC] -a NETWORK_ADDRESS -s SUBNET
                           [-v VIA] [-d DEV]
                           [-t {nat,local,unicast,broadcast,multicast,blackhole,unreachable,
↪throw,prohibit}]
                           [-b BIND] [-S {link,host,global}] [-n NAME] [-E]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -m METRIC, --metric METRIC
                        Metric of route, defaults to 100
  -a NETWORK_ADDRESS, --network_address NETWORK_ADDRESS
                        Destination network address, eg. 192.168.0.1
  -s SUBNET, --subnet SUBNET
                        Destination network subnet, eg. 24
  -v VIA, --via VIA     Gateway address. Routing means that (almost)
                        unmodified IP packet is sent over selected physical
                        link within physical packet addressed to gateway ---
                        therefore gateway must be directly reachable using
                        physical (e.g. MAC for Ethernet) address on given
                        interface --- this is by default ensured by kernel by
                        refusing to add route if gateway address is not in
                        subnet configured on given interface. Note that if the
                        gateway address is same as one of configured ip
                        addresses of EG own interfaces the --bind option will
                        be implied for this rule)
  -d DEV, --dev DEV     Device name (note that if this option is given --bind
                        is implied too), devices available in system {'eth0',
                        'lo'}
  -t {nat,local,unicast,broadcast,multicast,blackhole,unreachable,throw,prohibit}, --type {nat,
↪local,unicast,broadcast,multicast,blackhole,unreachable,throw,prohibit}
                        Type of route, defaults to 'unicast'
  -b BIND, --bind BIND  Bind route to specific interface. If rule is bound to
                        a specific interface system will automatically add or
                        remove this rule depending on the interface status.
                        Note that even if this option is not given manually it
                        may be implied based on --dev or --via options. Rules
                        not bound to specific interfaces are global and will
                        be applied only on EG startup or full reload of all
                        static routes.
  -S {link,host,global}, --scope {link,host,global}
                        Select rule scope
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset_edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

remove

Synopsis:

```
nm static_routing remove [-h] -c CONFIG -i ID [-n NAME] [-E]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        Name of config to be affected in currently edited
                        preset eg `lan1`, `lan2`. Configs are visible after
                        executing preset_print within edited preset.
```

(continues on next page)

(continued from previous page)

```
  -i ID, --id ID        ID of route to be deleted in currently edited preset
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset_edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

order

Synopsis:

```
nm static_routing order [-h] -c CONFIG -i ID -I ID [-n NAME] [-E]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        Name of config to be affected in currently edited
                        preset eg `lan1`, `lan2`. Configs are visible after
                        executing preset_print within edited preset.
  -i ID, --id ID        ID of first rule
  -I ID, --ID ID        ID of second rule
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset_edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

enable

Synopsis:

```
nm static_routing enable [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

disable

Synopsis:

```
nm static_routing disable [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

preset_create

This command creates a new preset in being_edited state. If `--source` option is given the new one will be a copy of the pointed source preset. This is atomic operation – even if more than one actor can access EG at the same time they will receive error if they try to perform conflicting atomic actions. Atomic operations are: preset selection, creating new preset, saving or making preset edited, reading all presets for backup or restoring all presets from backup.

Synopsis:

```
nm static_routing preset_create [-h] [-s SOURCE] name
```

Detailed description of positional arguments:

```
  name                  Name of the preset to create. No patterns allowed.
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -s SOURCE, --source SOURCE
                        Name of preset which shall be source of data for
                        preset being_edited. Shell glob patterns may be used,
                        but exactly one name must match pattern.
```

Examples:

create a new preset named `better_foo` as a copy of `good_old_foo`

```
  nm static_routing preset_create better_foo --source good_old_foo
```

preset_delete

This command removes existing preset. Before removal preset must be in being_edited state (see option `--make_edited`). Removed preset cannot be restored — use backup instead.

Synopsis:

```
nm static_routing preset_delete [-h] [-E] [name]
```

Detailed description of positional arguments:

```
  name                  Name of the preset being_edited which will be affected.
                        Shell glob patterns may be used but shall match exactly
                        one preset. If skipped this param defaults to '*' and
                        because normally only one preset is being edited, so this
                        argument is usally skipped.
```

Detailed description of named arguments:

```
  -h, --help          show this help message and exit
  -E, --make_edited   Mark saved preset as being edited one before performing
                      other actions, giving this option is the same as if
                      separate command `preset_edit` was executed by the user
                      just before this command --- it may fail if preset is
                      already being edited, or if another user tried to make
                      preset edited at the same time
```

Examples:

first make preset good_old_foo editable (this step may fail, which will prevent removal), then remove it

```
  nm static_routing preset_delete good_old_foo --make_edited
```

remove `better_foo` – this will fail if it is not being_edited

```
  nm static_routing preset_delete better_foo
```

remove preset which name ends with `foo` – this will fail if there is no such preset being edited already

```
  nm static_routing preset_delete '*foo'
```

preset_edit

This command can be applied on saved preset to make it editable. Editable presets cannot be accidentally selected, and saved presets cannot be accidentally edited or removed. Because of this allowing to edit selected preset makes no sense — the whole idea is to prevent potentially incomplete preset from being selected. To modify selected preset you need to make a copy of it (see command `preset_create -s`), then make changes, then save and select the copy. This is atomic operation – even if more than one actor can access EG at the same time they will receive error if they try to perform conflicting atomic actions. Atomic operations are: preset selection, creating new preset, saving or making preset edited, reading all presets for backup or restoring all presets from backup. Note that after starting the edition of the preset following editions are not atomic and if more than one actor performs them at the same time result might be unexpected — it is user responsibilty to ensure that second actor will not start modifying preset already in being_edited state before first actor finished his editions!

Synopsis:

```
nm static_routing preset_edit [-h] [name]
```

Detailed description of positional arguments:

```
  name        Name of saved preset. Shell glob patterns may be used but
              pattern must match exactly one preset name. If ommited defults
              to '*'
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

make preset good_old_foo editable

```
  nm static_routing preset_edit good_old_foo
```

preset_save

This command can be applied on being_edited preset to make it saved. Optionally a new name can given given to the preset during this operation. Editable presets cannot be accidentally selected, and saved presets cannot be accidentally edited or removed. This is atomic operation – even if more than one actor can access EG at the same time they will receive error if they try to perform conflicting atomic actions. Atomic operations are: preset selection, creating new preset, saving or making preset edited, reading all presets for backup or restoring all presets from backup.

Synopsis:

```
nm static_routing preset_save [-h] [-d DESTINATION] [name]
```

Detailed description of positional arguments:

```
  name                  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -d DESTINATION, --destination DESTINATION
                        Optional name under which currently being_edited
                        preset shall be saved, if not given current name of
                        preset being_edited will be used
```

Examples:

assuming only one preset is being edited currently save it under name `new_name`

```
  nm static_routing preset_save -d new_name
```

save preset which name starts with `my_` (there must be exactly one such preset in being edited state, but there may be other being edited presets whose names start differently

```
  nm static_routing preset_save 'my_*'
```

save the only being edited preset

```
  nm static_routing preset_save
```

preset_select

This command can be applied on saved preset to apply its contents as current configuration of EG. Any previously selected preset will cease to be selected anymore. This operation may require confirmation by the user after it is applied to prevent accidental loss of communication between user and EG. If such confirmation is not received it will be rolled back. If such confirmation is required, then operation starts at the moment of execution and ends at the moment it is rolled back or commited. Selected preset cannot be made editable. This is atomic operation – even if more than one actor can access EG at the same time they will receive error if they try to perform conflicting atomic actions. Atomic operations are: preset selection, creating new preset, saving or making preset edited, reading all presets for backup or restoring all presets from backup.

Synopsis:

```
nm static_routing preset_select [-h] [--unconditionally] [name]
```

Detailed description of positional arguments:

```
  name              Optional shell glob pattern to limit names of presets to
                    print (defaults to '*')
```

Detailed description of named arguments:

```
  -h, --help          show this help message and exit
  --unconditionally   Prevent connectivity checking after executing this
                      command (without this option user may be asked to confirm
                      that after command execution he has not lost connection
                      to the device, without such confirmation command would be
                      rolled back.)
```

Examples:

select and apply configuration in preset `location2_config`

```
  nm static_routing preset_select location2_config
```

preset_list

Lists existing presets (optionally limiting the result to presets matching glob pattern. Useful before executing commands to which concrete name is needed, or to check if preset is saved or being edited.

Synopsis:

```
nm static_routing preset_list [-h] [name]
```

Detailed description of positional arguments:

```
  name        Optional shell glob pattern to limit names returned (defaults to
              '*')
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

list all presets

```
  nm static_routing preset_list
```

list presets which names start with an ‘a’ and end with an ‘x’

```
  nm static_routing preset_list 'a*x'
```

preset_print

Prints contents of preset(s). For presets with multipart structure, if only one part is interesting output may be limited by giving the –part argument.

Synopsis:

```
nm static_routing preset_print [-h] [-p PART] [name]
```

Detailed description of positional arguments:

```
  name                  Optional shell glob pattern to limit names of presets
                        to print (defaults to '*')
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -p PART, --part PART  Part (at least container of rules) of preset which
                        shall be printed. Whole preset will printed if not
                        given.
```

Examples:

print contents of part `foo/bar/baz` in all presets

```
  nm static_routing preset_print --part foo/bar/baz
```

print contents of whole `foobar` preset

```
  nm static_routing preset_print foobar
```

5.1.14 promiscous_mode

Synopsis:

```
nm promiscous_mode [-h] interface {on,off}
```

Detailed description of positional arguments:

```
  interface   name of the interface
  {on,off}    turn on/off promiscous mode on selected interface
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

turn on promiscous mode on lan1

```
  nm promiscous_mode lan1 on
```

turn off promiscous mode on lan2

```
  nm promiscous_mode lan2 off
```

5.1.15 ids

Synopsis:

```
nm ids [-h] interface {enable,disable}
```

Detailed description of positional arguments:

```
  interface         name of the interface
  {enable,disable}  turn on/off IDS on selected interface
```

Detailed description of named arguments:

```
  -h, --help        show this help message and exit
```

Examples:

turn on IDS on lan1

```
nm ids lan1 enable
```

turn off IDS on lan2

```
nm ids lan2 disable
```

5.2 Device

This command has following subcommands:

```
   serial_console      enable/disable control of EG via serial port
   serial_get_config   get configuration of serial devices and stores it in a
                       file
   tpm_get             get registration_id and endorsement key from tpm
   user_password_hash  get/set user password hashes from /etc/shadow file
   azure_set_config    set Azure configuration
   azure_get_config    get Azure configuration and store in a file
   azure_set_string    set Azure configuration to use provided connection
                       string
   azure_set_tpm       set Azure configuration to use provided scope id and
                       registration id (TPM mode)
   azure_set_dps_x509  set Azure configuration to use provided scope id,
                       identity cert and identity pk
   azure_set_certificate
                       Add certificates for Iot Edge
   azure_configfile    Manage Azure IoT Edge config.
   azure_set_option    set Azure option
   motd_set            set message-of-the-day
   motd_get            get message-of-the-day
   get_config          get configuration and store in a file
   set_config          sets configuration and store in a file
   show                show configuration
   oss                 print open source software licenses
   user                Add, remove or list system users
   hostname            Set device hostname
   erase               Remove data from persistent storage
   logrotate           Change logrotation configuration
   get_serialnumber    get serial number of the device
   overcommit_memory   Change memory overcommit configuration.
   set_local_console   enable/disable login via local console
   get_local_console   check state of local linux console
   sshauth             Manage ssh authentication methods and public keys
   proxy               Set http/https proxy on system level (for logged in
                       users shells and for docker)
   login_timeout       Set automatic logout after idle timeout.
   localcertstore      Install certificate to local-device Trusted CA Store.
   get_logs            Prepare archive with logs
   smartems            Configure SmartEMS management service
   certificate         Manage security certificate used bye SmartEMS
   swupdate            Perform software update
   datetime            Configure date and time settings
```

5.2.1 serial_console

By enabling/disabling the console output one can change if (and if yes on which) serial port system will print console output and allow the user to use it with help of serial terminal emulator. Configuration of console output will be exported to global configuration file. Those changes can be permanent or temporary (regarding state after reboot). Hint: Enable or disable SysRq magic key functionality in the local_console settings for syskeys management.

Synopsis:

```
device serial_console [-h] {} {now,at_boot,both} {on,off}
```

Detailed description of positional arguments:

```
  {}                  which serial device will be affected (if this is empty
                      then the device is configured as not having serial
                      devices or the configuration file is missing)
  {now,at_boot,both}  if new state shall be immediate and/or applied also
                      after reboot
  {on,off}            turn the console on or off on selected device
```

Detailed description of named arguments:

```
  -h, --help          show this help message and exit
```

Examples:

after next (and later) reboots console will be active on serial port 1, current state will not be affected

```
  device serial_console 1 at_boot on
```

turn off console on serial port 0 immediately and keep it that way after reboot

```
  device serial_console 0 both off
```

turn on console at serial port 1 immediately, state after reboot will not be affected

```
  device serial_console 1 now on
```

5.2.2 serial_get_config

Gets configuration of serial ports and store it in a file `serial_config.json` (e.g. for backup purposes).

Synopsis:

```
device serial_get_config [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

save serial config to file `serial_config.json` and print it to the screen

```
  device serial_get_config ; cat ./serial_config.json
```

5.2.3 tpm_get

This function will get the endorsement key and registration id from built-in TPM module and print it out. Those keys can be used to provide provisioning configuration for Azure IoT Edge daemon. Note that TPM module may respond slowly.

Synopsis:

```
device tpm_get [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

the only way to call this command

```
  device tpm_get
```

5.2.4 user_password_hash

This command will get the users and their password hashes from /etc/shadow file and print it out or set a new ones provided in a config file.

This command has following subcommands:

```
    show             get user password hashes from /etc/shadow file
    set_config       set user password hashes to /etc/shadow file
```

Detailed description of named arguments:

```
  -h, --help         show this help message and exit
```

Examples:

```
device user_password_hash
```

show

Synopsis:

```
device user_password_hash show [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

set_config

Synopsis:

```
device user_password_hash set_config [-h] [-f FILENAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
                        path to a config file
```

5.2.5 azure_set_config

Restore Azure IoT Edge daemon configuration from a file. This is a permanent change (and will overwrite existing Azure configuration.)

Synopsis:

```
device azure_set_config [-h] [-f FILENAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
                        file with configuration data (default is
                        'azure_config.json')
```

Examples:

restore config from file `azure_config.json`

```
  device azure_set_config azure_config.json
```

Example content of config file:

```
{
   "azure":{
      "source":"manual",
      "device_connection_string":"connectionString",
      "hostname":"<ADD HOSTNAME HERE>"
   }
}
```

5.2.6 azure_get_config

Azure configuration (provisioning data) will be saved to json file named `azure_config.json`.

Synopsis:

```
device azure_get_config [-h] [--export-private-keys]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  --export-private-keys
```

Examples:

the only way to call this command

```
  device azure_get_config
```

5.2.7 azure_set_string

Configures Azure IoT Edge daemon to use connection string as a manual provisioning. Remember to use quotation marks for provided string (see examples).

Synopsis:

```
device azure_set_string [-h] connection_string
```

Detailed description of positional arguments:

```
  connection_string  connection string to be used
```

Detailed description of named arguments:

```
  -h, --help         show this help message and exit
```

Examples:

set connection string value to `Connection String`

```
  device azure_set_string 'Connection String'
```

5.2.8 azure_set_tpm

Configures Azure IoT Edge daemon to use TPM module for provisioning. This change is permament (performing configuration backup is up to the user)

Synopsis:

```
device azure_set_tpm [-h] id_scope registration_id
```

Detailed description of positional arguments:

```
  id_scope         ID scope from DPS
  registration_id  registration ID from TPM of the device
```

Detailed description of named arguments:

```
  -h, --help       show this help message and exit
```

Examples:

note that registrationIDFromTPMModule is device specific and can be obtained with command `device tpm_get`

```
  device azure_set_tpm scopeIDFromIoTHub registrationIDFromTPMModule
```

5.2.9 azure_set_dps_x509

Configures Azure IoT Edge daemon to use X.509 certificates provisioning method. Three arguments are needed for configuration: id scope from IoT Hub, certificate file and private key file. You need read permisions for these files. This change is permament (performing configuration backup is up to the user)

Synopsis:

```
device azure_set_dps_x509 [-h] -k IDENTITY_PK -p IDENTITY_CERT -i
                                ID_SCOPE
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -k IDENTITY_PK, --identity_pk IDENTITY_PK
  -p IDENTITY_CERT, --identity_cert IDENTITY_CERT
  -i ID_SCOPE, --id_scope ID_SCOPE
```

Examples:

```
device azure_set_dps_x509 -i scope_id_from_iot_hub -p /path/to/cert_file -k /path/to/private_key_
↪file
```

5.2.10 azure_set_certificate

Configures Azure IoT Edge daemon to use provided certificates. Device private key and device CA certificate will be used by device to prove its own identity. Device certificate needs to be signed by trust bundle certificate specific for IoT Edge scenario — public part of this trust bundle certificate is required on each device participating in the scenario too. Note that anyone knowing private key for device CA cert can impersonate this device in given IOT edge scenario

Synopsis:

```
device azure_set_certificate [-h] -t TRUST_BUNDLE_CERT -d
                                  DEVICE_CA_CERT -p DEVICE_CA_PK
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -t TRUST_BUNDLE_CERT, --trust_bundle_cert TRUST_BUNDLE_CERT
  -d DEVICE_CA_CERT, --device_ca_cert DEVICE_CA_CERT
  -p DEVICE_CA_PK, --device_ca_pk DEVICE_CA_PK
```

Examples:

all 3 files must be readable for you

```
  device azure_set_certificate -t path/to/trust_bundle_cert -d path/to/device_ca_cert -p path/to/
↪device_ca_private_key
```

5.2.11 azure_configfile

Get full Azure IoT Edge configuration in toml file or upload preconfigured file. Notice: Exported file contains all settings, but some of them are ignored during import (see the details in description of import)

This command has following subcommands:

```
    export          Get toml with Azure configuration. All entries which are
                    set in Azure config file are present, so user can use for
                    analysis or to test it outside of Edge Gateway
    import          Set Azure config based on given toml file.
```

Detailed description of named arguments:

```
  -h, --help        show this help message and exit
```

Examples:

```
device azure_configfile
```

export

Synopsis:

```
device azure_configfile export [-h] [-f FILENAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
                        file to write to (default is 'iotedge.toml')
```

import

This method is designed to allow import of Azure config prepared on machine other than Edge Gateway, therefore parameters which are handled specially for Edge Gateway will be ignored. This affects especially hostname and certificates, but also some paths which shall not be modified. Response to this command will inform user if any entries were ignored. To see exactly what was ignored you can run export and compare your input file with reexported contents. If you want to export/import whole Azure configuration between Edge Gateways in simpler way use azure_get_config/azure_set_config commands instead of azure_configfile export/import.

Synopsis:

```
device azure_configfile import [-h] -f FILENAME
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
```

5.2.12 azure_set_option

Change selected entry in IoT Edge TOML config. This option takes a string as a `topic` parameter which indicates topic-key in TOML tree to be changed. If selected topic-key does not exist system will create it in config file but if it does exist this command will change its value to the one specified in `entry` parameter. The `entry` parameter can be used more than once. When `topic` is not given entries are treated as global key-value pairs. This command does not verify if topic-key name and entry is reasonable and understood by Azure, but some options are protected and cannot be changed this way if it would not make sense in Edge Gateway (for example certificate paths are predefined for Edge Gateways, instead of changing those paths in Azure config you shall use azure_set_certificate command).

Synopsis:

```
device azure_set_option [-h] [-t TOPIC] -e ENTRY [-d]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -t TOPIC, --topic TOPIC
                        topic to setup
  -e ENTRY, --entry ENTRY
                        entry being key-value pair to be set, note that TOML
                        will NOT interpret its type. You need to use proper
                        quotation as in examples above.
  -d, --dry_run         Do not set option, but show partial TOML which would
                        be added to config
```

Examples:

change attestation method to tpm

```
device azure_set_option -t  provisioning.attestation -e 'method = "tpm"'
```

show partial TOML to add topic f.b.z with value being list of two strings

```
device azure_set_option -t f.b.z -e 'l = ["a", "1"]' -d
```

show partial TOML to add topic f.b.z with value being list of a string and a number

```
device azure_set_option -t f.b.z -e 'l = ["a", 1]' -d
```

add topic f.b.z with two entries

```
device azure_set_option -t f.b.z -e 'x = "a"' -e 'y = "b"'
```

5.2.13 motd_set

set new welcome-banner (MotD), shown after logging in

Synopsis:

```
device motd_set [-h] (-f FILE | -t TEXT)
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILE, --file FILE  set new MotD from file
  -t TEXT, --text TEXT  set new MotD from plain-text
```

5.2.14 motd_get

get current welcome-banner (MotD), shown after logging in

Synopsis:

```
device motd_get [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.2.15 get_config

Whole device configuration will be saved to json file named `device_config.json`.

Synopsis:

```
device get_config [-h] [--export-private-keys]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  --export-private-keys
                        Governs export of private keys and user password
                        hashes
```

Examples:

the only way to call this command

```
device get_config
```

5.2.16 set_config

Whole device configuration will be restored from a file.

Synopsis:

```
device set_config [-h] [-f FILENAME] [--unconditionally]
```

Detailed description of named arguments:

```
-h, --help            show this help message and exit
-f FILENAME, --filename FILENAME
                      file with configuration data (default is '{}')
--unconditionally     Prevent connectivity checking after executing this
                      command (without this option user may be asked to
                      confirm that after command execution he has not lost
                      connection to the device, without such confirmation
                      command would be rolled back.)
```

Examples:

restore config from file `device_config.json`

```
device set_config --filename device_config.json
```

Example JSON file:

```
{
    "azure": {
        "source": "manual",
        "device_connection_string": "<ADD DEVICE CONNECTION STRING HERE>",
        "hostname": "<ADD HOSTNAME HERE>"
    },
    "serial": {
        "serial0": {
            "console_output": {
                "at_startup": true,
                "currently": true
            }
        },
        "serial1": {
            "console_output": {
                "at_startup": false,
                "currently": false
            }
        }
    },
    "network": {
        "lan2": {
            "dhcp": true,
            "ignore_default_route": false,
            "current_ip": "192.168.1.95",
            "current_subnet": "24",
            "current_gateway": "192.168.1.1",
            "current_dns": "1.1.1.1"
        },
        "lan1": {
```

(continues on next page)

```
            "dhcp": false,
            "ip": [
                "192.168.2.1"
            ],
            "subnet": [
                "24"
            ],
            "gateway": null,
            "dns": null,
            "current_ip": "192.168.2.1",
            "current_subnet": "24",
            "current_gateway": null,
            "current_dns": null
        }
    },
    "static_routing": {
        "enabled": false,
        "selected": "disabled",
        "saved": {
            "disabled": {}
        },
        "edited": {}
    },
    "firewall": {
        "enabled": false,
        "selected": "allow_all",
        "edited": {},
        "saved": {
            "allow_all": {
                "inet": {
                    "filter": {
                        "output": {
                            "policy": "accept"
                        },
                        "input": {
                            "policy": "accept"
                        },
                        "forward": {
                            "policy": "accept"
                        }
                    }
                },
                "ip": {
                    "nat": {
                        "postrouting": {
                            "policy": "accept"
                        },
                        "prerouting": {
                            "policy": "accept"
                        }
                    }
                },
                "notes": "This is the minimum configuration of firewall. This preset cannot be␣
↪modified or removed.\n\nThis configuration does not drop or alter any packets."
            },
            "disabled": {
                "inet": {
                    "filter": {
                        "output": {
```


(continues on next page)

(continued from previous page)

```
                            "policy": "accept"
                        },
                        "input": {
                            "policy": "accept"
                        },
                        "forward": {
                            "policy": "accept"
                        }
                    }
                },
                "ip": {
                    "nat": {
                        "postrouting": {
                            "policy": "accept"
                        },
                        "prerouting": {
                            "policy": "accept"
                        }
                    }
                },
                "notes": "This preset disables firewall. This preset cannot be modified or removed.
↪\n\nThere is special closure/disable part in this preset which totally flushes out\nfirewall␣
↪tables. Effect is very similar to allow all, but conceptually the\ndifference is that here we␣
↪ensure that firewall is really doing nothing\n(theoretically allow a ll could be doing actions␣
↪like categorizing and counting\npackets).",
                "closure": {
                    "disable": {
                        "contents": "flush ruleset",
                        "notes": "Removes all rules from firewall"
                    }
                }
            },
        }
    },
    "ovpn": {}
}
```

5.2.17 Set datetime using `date` binary

It is possbile to set a system time in the device locally using 'date' command. Please note that Edge Gateway is configured with NTP enabled in factory default which will overwrite manually set datetime values.

```
Usage: date [OPTION]... [+FORMAT]
  or:  date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
Display date and time in the given FORMAT.
With -s, or with [MMDDhhmm[[CC]YY][.ss]], set the date and time.
Mandatory arguments to long options are mandatory for short options too.
  -d, --date=STRING          display time described by STRING, not 'now'
      --debug                annotate the parsed date,
                              and warn about questionable usage to stderr
  -f, --file=DATEFILE        like --date; once for each line of DATEFILE
  -I[FMT], --iso-8601[=FMT]  output date/time in ISO 8601 format.
                               FMT='date' for date only (the default),
                               'hours', 'minutes', 'seconds', or 'ns'
                               for date and time to the indicated precision.
                               Example: 2006-08-14T02:34:56-06:00
  --resolution               output the available resolution of timestamps
                               Example: 0.000000001
```

(continues on next page)

(continued from previous page)

```
  -R, --rfc-email             output date and time in RFC 5322 format.
                                Example: Mon, 14 Aug 2006 02:34:56 -0600
      --rfc-3339=FMT          output date/time in RFC 3339 format.
                                FMT='date', 'seconds', or 'ns'
                                for date and time to the indicated precision.
                                Example: 2006-08-14 02:34:56-06:00
  -r, --reference=FILE        display the last modification time of FILE
  -s, --set=STRING            set time described by STRING
  -u, --utc, --universal      print or set Coordinated Universal Time (UTC)
      --help        display this help and exit
      --version     output version information and exit
```

Example usage:

```
date -s '1984-01-04 09:33:00'
```

5.2.18 show

Whole device configuration will be printed out. Note that this is quite verbose.

Synopsis:

```
device show [-h] [--export-private-keys]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  --export-private-keys
```

Examples:

the only way to call this command

```
  device show
```

5.2.19 oss

Retrieves information about installed packages and their licenses.

Synopsis:

```
device oss [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

the only way to call this command

```
  device oss
```

5.2.20 user

Add, remove or list system users. User home directories are created automatically with paths specific for the group of a user. For the `user` group account path to home is `/home/ro_users/{usrnm}` where `{usrnm}` is the username. For the `admin` group account path to home is `/home/admins/{admnm}` where `{admnm}` is the username.

This command has following subcommands:

```
  add               Add system users
  remove            Remove system users
  list              List system users
```

Detailed description of named arguments:

```
  -h, --help        show this help message and exit
```

Examples:

adds read only user `user_ro` with password 12345

```
  device user add -g user -p 12345 -u user_ro
```

adds admin user

```
  device user add -g admin -p 12345 -u admin
```

remove user but leave his home directory on disk

```
  device user remove -u user_ro
```

remove user and all of his home files

```
  device user remove -u admin -d
```

add

Synopsis:

```
device user add [-h] -u USERNAME -g {admin,user} -p PASSWORD
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -u USERNAME, --username USERNAME
                        Login name (which is a unique identifier in the
                        system) of a user to be added.
  -g {admin,user}, --group {admin,user}
                        Group of a user determines permissions in the system
                        and is mandatory when adding new user. Users of group
                        `admin` have full access configuration management,
                        users of group `user` can only execute commands which
                        do not change configuration. To choose account group,
                        available options are `user` with read only access to
                        CLI and `admin` with full access to CLI.
  -p PASSWORD, --password PASSWORD
                        Password is mandatory when adding user.
```

remove

Synopsis:

```
device user remove [-h] -u USERNAME [-d]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -u USERNAME, --username USERNAME
                        Login name (which is a unique identifier in the
                        system) of a user to be removed.
  -d, --deletehome      Causes user home directory and its contents to be
                        deleted.
```

list

Synopsis:

```
device user list [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.2.21 hostname

Sets device hostname, this will also set hostname in IoT Edge daemon config. Note that it may brake Azure configuration which could be depending on that name.

Synopsis:

```
device hostname [-h] name
```

Detailed description of positional arguments:

```
  name
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

change own hostname to eg210 and check that it can be pinged

```
  device hostname eg210; ping -c 1 eg210
```

5.2.22 erase

Allows to perform full factory reset of device. All files created after installation (including user files, dockers and config for services) will be removed.

Synopsis:

```
device erase [-h] {everything}
```

Detailed description of positional arguments:

```
  {everything}  what to erase
```

Detailed description of named arguments:

```
  -h, --help    show this help message and exit
```

Examples:

perform factory reset

```
  device erase everything
```

5.2.23 logrotate

Changes when log files will be renamed and removed

This command has following subcommands:

```
    set       Change config for logrotate service
    show      Show current config
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

new log file will be created after current grows above 100MB or new day starts, keep only last 7 log files on disk

```
  device logrotate set --size 100 --rotate 7 --period daily
```

display current logrotate configuration

```
  device logrotate show
```

set

Synopsis:

```
device logrotate set [-h] -p {hourly,daily,weekly,monthly} -r ROTATE -s
                         SIZE
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -p {hourly,daily,weekly,monthly}, --period {hourly,daily,weekly,monthly}
                        How often log files are rotated.
  -r ROTATE, --rotate ROTATE
                        Logrotate rotates the log files that many times before
                        removal. If it's set to 0, old versions are removed
                        rather than rotated.
  -s SIZE, --size SIZE  Maximum size of the log file in megabytes. Logrotate
                        rotates the log at selected period, but when the log
                        file reaches it's maximum size, logrotate rotates log
                        despite the period.
```

show

Synopsis:

```
device logrotate show [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.2.24 get_serialnumber

Serial number of the device will be printed out.

Synopsis:

```
device get_serialnumber [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

the only way to call this command

```
  device get_serialnumber
```

5.2.25 overcommit_memory

This enables or disables overcommitting of the memory. When disabled, the machine will not run programs that would exceed available memory. When enabled, OOM killer might kill programs that it finds suitable to kill.

Synopsis:

```
device overcommit_memory [-h] {enable,disable,default,status}
```

Detailed description of positional arguments:

```
  {enable,disable,default,status}
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
```

Examples:

```
device overcommit_memory enable
device overcommit_memory disable
device overcommit_memory status
device overcommit_memory default
```

5.2.26 set_local_console

This enables or disables linux local console (TTY) *** IMPORTANT - this command, might lock you out.

Synopsis:

```
device set_local_console [-h] [-l {enable,disable}]
                               [-s {enable,disable}]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -l {enable,disable}, --login {enable,disable}
                        Enable or disable linux local login
  -s {enable,disable}, --syskeys {enable,disable}
                        Enable or disable Ctrl-Alt-Del, SysRq keys action.
                        This option enables or disables SysRq to both
                        consoles: serial (COM1) and local (TTY)
```

5.2.27 get_local_console

Synopsis:

```
device get_local_console [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.2.28 sshauth

Allows to manage if password and public key based authentication is available for ssh connecitons. At least one method must be allowed. If password based authentication is going to be turned off, then at least admin account must have at least one valid public key present in "~/.ssh/authorized_keys".

This command has following subcommands:

```
    set_config          Replace ssh config and user keys with new values
    get_config          Return current ssh config
    set                 Change allowed types of ssh authentication
    show                Show current ssh authentication configuration
    list_publickeys     Display content of ~/.ssh/authorized_keys for
                        currently logged in user
    add_publickey       Adds a public key from a keypair for selected user to
                        authorized_keys. If you want to update comment to
                        already existing key first you have removed it with
                        remove_key option.
    remove_key          Remove key with given index for currently logged in
                        user
    maxsessions         Specifies the maximum number of open shell, login or
                        subsystem (e.g. sftp) sessions permitted per network
                        connection.
    maxstartups         Specifies the maximum number of concurrent
                        unauthenticated connections to the SSH daemon.
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
```

Examples:

allow key based authentication but disable password based one

```
device sshauth set --public_key_auth on --password_auth off
```

disable password auth (leave key based auth in same state as it was before — this command will fail if key based authentication was turned off already)

```
device sshauth set -p off
```

display current ssh authenctication configration

```
device sshauth show
```

set_config

This will first validate that at least one authentication method is enabled in incoming config, if yes then proceed to replace users ssh public keys. In case of failure for any user, allowed authentication methods will be set as requested but password based authentication will be enabled always in case admin user authorized_keys will be empty, to prevent total lock out from device. Keys of users not listed in the json will not be modified — to remove keys of users they need to be present in json file with empty list of keys. Warning: in case of issues with keys in json file it may cause state where some users have changed keys, and some user have old keys — please analyze failures of this command carefully.

Synopsis:

```
device sshauth set_config [-h] [-f FILENAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
                        file with configuration data (default is
                        'ssh_config.json')
```

get_config

Synopsis:

```
device sshauth get_config [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

set

Synopsis:

```
device sshauth set [-h] [-k {on,off}] [-p {on,off}]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -k {on,off}, --public_key_auth {on,off}
  -p {on,off}, --password_auth {on,off}
```

show

Synopsis:

```
device sshauth show [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

list_publickeys

Synopsis:

```
device sshauth list_publickeys [-h] [-u USERNAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -u USERNAME, --username USERNAME
```

add_publickey

Synopsis:

```
device sshauth add_publickey [-h] -f FILENAME [-u USERNAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
  -u USERNAME, --username USERNAME
```

remove_key

Synopsis:

```
device sshauth remove_key [-h] -i INDEX [-u USERNAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -i INDEX, --index INDEX
  -u USERNAME, --username USERNAME
```

maxsessions

Synopsis:

```
device sshauth maxsessions [-h] sessions
```

Detailed description of positional arguments:

```
  sessions
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

maxstartups

Synopsis:

```
device sshauth maxstartups [-h] startups
```

Detailed description of positional arguments:

```
  startups
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.2.29 proxy

This command has following subcommands:

```
    add                 Add or overrwrite proxy or proxies
    del                 Delete proxies
    set_config          Replace proxy config with new one
    get_config          Return current proxy config (except for docker-
                        compose)
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
```

add

You may give http and/or https proxy parameters. This command will add them (or overwrite if any is already configured) for azure daemons, docker and shell environment as well as for docker-compose (Note: docker-compose stores its proxy settings in separate physical location and this forces docker- compose proxy to be kept in separate subsection of EG global json config). If you give proxy only for one protocol (http/https) and the other is already configured, this other will not be removed. For individual users' environments, logging in again is needed to notice configuration change. Immediately after storing new proxy in the configurations response will be sent back to the user and daemons which use proxy (docker and containers) will be restarted. This restart of deamons can take considerable amount of time (and potentially disrupt processing done by containers).

Synopsis:

```
device proxy add [-h] [-p HTTP] [-s HTTPS] [-n]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -p HTTP, --http HTTP  set HTTP-Proxy to http://<SERVER>:<PORT> (e.g.
                        http://192.168.123.254:8080 )
  -s HTTPS, --https HTTPS
                        set HTTPS-Proxy to https://<SERVER>:<PORT> (e.g.
                        https://192.168.123.254:8080 )
  -n, --no_reload       Change configuration, but do not reload it immediately
                        in all daemons. If you give this option new proxy will
                        be used by docker containers after next reboot of the
                        system or next restart of docker subsystem (whichever
```

(continues on next page)

(continued from previous page)

```
                        comes first). To manually enforce restart of docker
                        execute command `docker-config apply`.
```

del

Delete http and/or https proxies

Synopsis:

```
device proxy del [-h] [--http] [--https]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
  --http      delete http proxy
  --https     delete https proxy
```

set_config

This will add/remove system and docker daemon proxies to match contents of given config. This will not change docker-compose proxy settings! Docker and containers will be restarted according to boolean entry 'reload_daemons' which can optionally be present next to the object proxy_servers in json file used with this command. If that entry is missing its value is assumed to be true.

Synopsis:

```
device proxy set_config [-h] [-f FILENAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
                        file with configuration data (default is
                        'proxy_config.json')
```

get_config

This will return partial EG config for proxy only. Note that because docker- compose stores its proxy settings in separate place you would need to use command `docker-config compose get-config` to get partial config containing proxy of docker-compose.

Synopsis:

```
device proxy get_config [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.2.30 login_timeout

This will set a system-wide auto-logout policy. Every user, after N-seconds of idle will be logged off.

This command has following subcommands:

```
   set       Set the value for automatic idle logout
   get       Get the value for automatic idle logout
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

set

Configure every user login to be closed after idle time

Synopsis:

```
device login_timeout set [-h] -s SECONDS
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -s SECONDS, --seconds SECONDS
                        Amount of idle seconds for automatic logout (0 means
                        infinity, values 1 to 9 are rejected)
```

get

Read All-Users default idle logout time

Synopsis:

```
device login_timeout get [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.2.31 localcertstore

This command has following subcommands:

```
   install   Install new CRT
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

install

Synopsis:

```
device localcertstore install [-h] -f FILE
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILE, --file FILE  Path to CRT file
```

5.2.32 get_logs

Prepares archive `logs.zip` with currently existing logs.

Synopsis:

```
device get_logs [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

the only way to call this command

```
  device get_logs
```

5.2.33 smartems

This command has following subcommands:

```
    config              Change config for auto update service
    set_config          Replace current Smart EMS configuration
    show                Show current config
    check               Trigger immediate check for new Smart EMS management
                        commands
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
```

Examples:

display current configuration

```
  device smartems show
```

immediately connect to SmartEMS to check for any management commands

```
  device smartems check
```

Other examples:

```
device smartems config --username user --password pass --url url
device smartems config --username user --password pass --url url --vcc_api_endpoint
```

config

Synopsis:

```
device smartems config [-h] -u USERNAME -p PASSWORD -U URL [-s]
                       [-i INTERVAL] [--vcc_api_endpoint]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -u USERNAME, --username USERNAME
                        Username for Smart EMS
  -p PASSWORD, --password PASSWORD
                        Password for Smart EMS
  -U URL, --url URL     URL for Smart EMS
  -s, --skip            Skip checking SSL certificate for Smart EMS.DANGER!
                        This allows man-in-the-middle attacks.
  -i INTERVAL, --interval INTERVAL
                        Period between two checks, in seconds. The minimum is
                        10s.
  --vcc_api_endpoint    If present, this API endpoint will be set -
                        /api/edgegatewayvcc/configuration. Otherwise, this one
                        will be used - /api/edgegateway/configuration. If your
                        EdgeGateway is connected to VPN CC use this option.
```

set_config

Synopsis:

```
device smartems set_config [-h] [-f FILENAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
                        file with configuration data (default is
                        'smart_ems.json')
```

show

Synopsis:

```
device smartems show [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

check

This command will trigger an immediate connection to the SmartEMS and perform any updates requested by SmartEMS (it may be config and or software update) or provide information requested by SmartEMS (like current configuration). If software update is requested then EG will download package first — depending on connection speed this may consume bigger amount of time. This command times out after five minutes but even in such case download is still being performed in the background. The new firmware package is being stored in /tmp and has .swu extension, so you can check in that directory if it is growing. After firmware is downloaded it will be automatically installed and system will reboot.

Synopsis:

```
device smartems check [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.2.34 certificate

This command has following subcommands:

```
    add               Add new certificate
    delete            Remove current certificate
    show              Display current certificate
```

Detailed description of named arguments:

```
  -h, --help          show this help message and exit
```

add

Synopsis:

```
device certificate add [-h] -c CERTIFICATE
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -c CERTIFICATE, --certificate CERTIFICATE
                        Filename with certificate
```

delete

Synopsis:

```
device certificate delete [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

show

Synopsis:

```
device certificate show [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.2.35 swupdate

Synopsis:

```
device swupdate [-h] -f FILENAME
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
                        SWU file
```

Examples:

perform software update with selected .swu file

```
  device swupdate -f/--filename
```

5.2.36 datetime

This command has following subcommands:

```
    set_config          Replace ntp config with new one
    get_config          Return current ntp and timezone config
    show                Show current datetime status
    set_ntp             Change NTP configuration
    enable              Enable NTP service
    disable             Enable NTP service
    list_timezones      List all avaible timezones
    set_timezone        List all avaible timezones
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
```

Examples:

display current NTP status

```
  device datetime show
```

enable NTP service

```
  device datetime enable
```

disable NTP service

```
  device datetime disable
```

list supported timezones

```
  device datetime list_timezones
```

set timezone to Berlin time

```
  device datetime set_timezone Europe/Berlin
```

set NTP to use provided servers

```
device datetime set_ntp --server "time1.ntp time2.ntp"
```

set primary and fallback servers for NTP

```
device datetime set_ntp --server "time1.ntp time2.ntp" --fallback- servers "time3.ntp time4.ntp"
```

set minimum and maximum intervals (in seconds) beetwen two synchronization events

```
device datetime set_ntp --server time1.ntp --interval-minimum 16 --interval-maximum 32
```

set_config

This will set configuration of ntp and select timezone.

Synopsis:

```
device datetime set_config [-h] [-f FILENAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
                        file with configuration data (default is
                        'datetime_config.json')
```

get_config

Synopsis:

```
device datetime get_config [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

show

Synopsis:

```
device datetime show [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

set_ntp

Synopsis:

```
device datetime set_ntp [-h] -s SERVER [-f FALLBACK_SERVERS]
                            [-i INTERVAL_MINIMUM] [-I INTERVAL_MAXIMUM]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -s SERVER, --server SERVER
                        IP address or hostname for main NTP servers, this
                        paramter accept multiple servers separated by spaces.
                        If multiple servers are provided thay have to be
                        enclosed within quotation marks
  -f FALLBACK_SERVERS, --fallback-servers FALLBACK_SERVERS
                        IP address or hostname for fallback server, this
                        paramter accepts multiple servers separated by spaces.
                        Those servers will be used to synchronize time when
                        main NTP server is inaccessible. Defaults to:
                        time1.google.com time2.google.com time3.google.com
                        time4.google.com. If multiple servers provided thay
                        have to be enclosed with quotation mark.
  -i INTERVAL_MINIMUM, --interval-minimum INTERVAL_MINIMUM
                        Minimum time in seconds between two NTP messages.
                        Defaults to 32 Minimum value is 16, maximum value is
                        2048. Minimum value can not be grater then maximum
                        interval.
  -I INTERVAL_MAXIMUM, --interval-maximum INTERVAL_MAXIMUM
                        Maximum time in seconds between two NTP messages.
                        Defaults to 2048. Minimum value is 16, maximum value
                        is 2048. Maximum time can not be lesser than minimum
                        time.
```

enable

Synopsis:

```
device datetime enable [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

disable

Synopsis:

```
device datetime disable [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

list_timezones

Synopsis:

```
device datetime list_timezones [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

set_timezone

Synopsis:

```
device datetime set_timezone [-h] {}
```

Detailed description of positional arguments:

```
  {}          Set timezone
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.3 Firewall

This command has following subcommands:

```
    default             select and enable default firewall config (based on
                        preset default
    enable              enable firewall using configuration selected earlier
                        by preset_select or set_config
    disable             disables firewall
    set_config          restore firewall configuration from a json file
    get_config          get configuration to firewall_config.json file
    show                show firewall configuration in terse way
    commit              affirm that current state of firewall is correct,
                        '(prevents rollback of yet uncommited actions)
    cleanup             remove traces of uncommited transaction without
                        touching rest of the config
    reload              reload configuration from scratch, as would be done
                        after reboot
    preset              Manage firewall preset states (e.g. mark preset as
                        being edited)
    modify              Modify preset in being edited state
    print               print contents of preset being edited (whole or just a
                        part)
```

Unrelated commands examples:

```
fw default
fw enable
fw disable
fw set_config --filename firewall.config
fw get_config
fw preset_create my_new_preset -s allow_all
fw modify set_policy -r -p inet drop
fw modify set_policy -p inet/filter/output accept
fw print
fw modify erase -p inet/filter/common/mgmtd/cli_allow_null_22
fw modify erase -p inet/filter/input/mgmtd/cli_allow_lan1_23
```

Multicommand example in which we create new preset (based on default one) with rule that presents virtual IP 192.168.2.202 visible on the lan1 side which is redirected to hidden private IP 192.168.1.102. We also disable any forwarding except the connections originating from the hidden private IP (for such connections traffic in both ways will be accepted thanks to preexisting connection tracking rule in default preset — we know about that rule because we have printed preset just after creating it). After we add our new rules we again print the modified part, then save, select and enable our preset:

```
fw preset create virt_ip_on_lan1 -s default
fw print
fw modify nat_n_on_n add from_lan1to2 --public_interface lan1 --ip_public 192.168.2.202 --ip_
↪private 192.168.1.102
fw modify set_policy -p inet/filter/forward drop
fw modify forward -s 192.168.1.102 -v accept
fw print -p ip
fw preset save
fw preset select virt_ip_on_lan1
fw enable
```

5.3.1 default

Select a default preset (currently allow_management) and enables it All firewall rules that are currently set will be cleared. Also, the firewall will be enabled and set to automatically start on boot.

Synopsis:

```
fw default [-h] [--unconditionally]
```

Detailed description of named arguments:

```
  -h, --help          show this help message and exit
  --unconditionally   Prevent connectivity checking after executing this
                      command (without this option user may be asked to confirm
                      that after command execution he has not lost connection
                      to the device, without such confirmation command would be
                      rolled back.)
```

Examples:

the only way to call this command

```
  fw default
```

5.3.2 enable

Loads firewall rules into the kernel immediately and after reboot. Before enabling firewall a preset must be selected (`default` preset is selectd initially after factory reset).

Synopsis:

```
fw enable [-h] [--unconditionally]
```

Detailed description of named arguments:

```
  -h, --help          show this help message and exit
  --unconditionally   Prevent connectivity checking after executing this
                      command (without this option user may be asked to confirm
                      that after command execution he has not lost connection
                      to the device, without such confirmation command would be
                      rolled back.)
```

Examples:

the only way to call this command

```
  fw enable
```

5.3.3 disable

Makes firewall incative (it will not touch any packets) immediatelly and after reboot

Synopsis:

```
fw disable [-h] [--unconditionally]
```

Detailed description of named arguments:

```
  -h, --help          show this help message and exit
  --unconditionally   Prevent connectivity checking after executing this
                      command (without this option user may be asked to confirm
                      that after command execution he has not lost connection
                      to the device, without such confirmation command would be
                      rolled back.)
```

Examples:

the only way to call this command

```
  fw disable
```

5.3.4 set_config

Restores presets from json file. Note that factory presets cannot be modified with this command.

Synopsis:

```
fw set_config [-h] --filename FILENAME [--unconditionally]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  --filename FILENAME   json filename
  --unconditionally     Prevent connectivity checking after executing this
                        command (without this option user may be asked to
                        confirm that after command execution he has not lost
                        connection to the device, without such confirmation
                        command would be rolled back.)
```

Examples:

```
fw set_config --filename firewall_config.json
```

5.3.5 get_config

Stores firewall config (including all presets) in file `firewall_config.json`

Synopsis:

```
fw get_config [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

the only way to call this command

```
fw get_config
```

5.3.6 show

Shows firewall configuration in terse way (i.e. notes will be hidden, use `{SUBPARSER_PRESET} print` with name of selected preset to see also notes

Synopsis:

```
fw show [-h]
```

Detailed description of named arguments:

```
-h, --help  show this help message and exit
```

Examples:

the only way to call this command

```
fw show
```

5.3.7 commit

If transaction has been started and not rolled backed yet (currently transaction timeout is always 30 seconds, but it will be configurable in the future) this command can be executed to prevent rollback and finish transaction immediately.

Synopsis:

```
fw commit [-h]
```

Detailed description of named arguments:

```
-h, --help  show this help message and exit
```

Examples:

the only way to call this command

```
fw commit
```

5.3.8 cleanup

If transaction was not finished by commit nor rollback and firewall configuration management daemon was restarted (e.g. due to power failure or crash) unclean state on disk may prevent any other transactions to be performed indefinitely. This command allows to recover from such state. Cleanup will first try to commit transaction, and if it fails it will remove leftovers. Restoring config from backup after using cleanup is reccomended.

Synopsis:

```
fw cleanup [-h]
```

Detailed description of named arguments:

```
-h, --help  show this help message and exit
```

Examples:

the only way to call this command

```
fw cleanup
```

5.3.9 reload

Reloads firewall configuration. If firewall has some state (e.g. is tracking existing connections) it will be reset.

Synopsis:

```
fw reload [-h]
```

Detailed description of named arguments:

```
-h, --help  show this help message and exit
```

Examples:

the only way to call this command

```
fw reload
```

5.3.10 preset

Manages firewall prests — creation, marking as being edited, saving, and most importantly selecting the preset being used by EG for actual firewall configuration.

This command has following subcommands:

```
  create              create new preset in being_edited state
  delete              delete preset currently being_edited
  edit                mark saved preset as being_edited (allows to change
                      its contents, but prevents selecting it as current
                      one)
  save                mark preset being_edited as saved
  select              select saved preset for use
  list                list existing presets (both saved and being_edited)
  print               print contents of preset(s) (whole or just a part)
```

Detailed description of named arguments:

```
-h, --help            show this help message and exit
```

Examples:

selects allow_all preset for firewall configuration, if firewall was already enabled, new preset will be immediatelly applied

```
fw preset select allow_all
```

creates new being edited preset `my_own` as copy of `allow_management` preset

```
fw preset create my_own -s allow_management
```

create

This command creates a new preset in being_edited state. If `--source` option is given the new one will be a copy of the pointed source preset. This is atomic operation – even if more than one actor can access EG at the same time they will receive error if they try to perform conflicting atomic actions. Atomic operations are: preset selection, creating new preset, saving or making preset edited, reading all presets for backup or restoring all presets from backup.

Synopsis:

```
fw preset create [-h] [-s SOURCE] name
```

Detailed description of positional arguments:

```
  name                  Name of the preset to create. No patterns allowed.
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -s SOURCE, --source SOURCE
                        Name of preset which shall be source of data for
                        preset being_edited. Shell glob patterns may be used,
                        but exactly one name must match pattern.
```

Examples:

create a new preset named `better_foo` as a copy of `good_old_foo`

```
  fw preset create better_foo --source good_old_foo
```

delete

This command removes existing preset. Before removal preset must be in being_edited state (see option `--make_edited`). Removed preset cannot be restored — use backup instead.

Synopsis:

```
fw preset delete [-h] [-E] [name]
```

Detailed description of positional arguments:

```
  name                  Name of the preset being_edited which will be affected.
                        Shell glob patterns may be used but shall match exactly
                        one preset. If skipped this param defaults to '*' and
                        because normally only one preset is being edited, so this
                        argument is usally skipped.
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -E, --make_edited     Mark saved preset as being edited one before performing
                        other actions, giving this option is the same as if
                        separate command `edit` was executed by the user just
                        before this command --- it may fail if preset is already
                        being edited, or if another user tried to make preset
                        edited at the same time
```

Examples:

first make preset good_old_foo editable (this step may fail, which will prevent removal), then remove it

```
fw preset delete good_old_foo --make_edited
```

remove `better_foo` – this will fail if it is not being_edited

```
fw preset delete better_foo
```

remove preset which name ends with `foo` – this will fail if there is no such preset being edited already

```
fw preset delete '*foo'
```

edit

This command can be applied on saved preset to make it editable. Editable presets cannot be accidentally selected, and saved presets cannot be accidentally edited or removed. Because of this allowing to edit selected preset makes no sense — the whole idea is to prevent potentially incomplete preset from being selected. To modify selected preset you need to make a copy of it (see command `create -s`), then make changes, then save and select the copy. This is atomic operation – even if more than one actor can access EG at the same time they will receive error if they try to perform conflicting atomic actions. Atomic operations are: preset selection, creating new preset, saving or making preset edited, reading all presets for backup or restoring all presets from backup. Note that after starting the edition of the preset following editions are not atomic and if more than one actor performs them at the same time result might be unexpected — it is user responsibilty to ensure that second actor will not start modifying preset already in being_edited state before first actor finished his editions!

Synopsis:

```
fw preset edit [-h] [name]
```

Detailed description of positional arguments:

```
  name        Name of saved preset. Shell glob patterns may be used but
              pattern must match exactly one preset name. If ommited defults
              to '*'
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

Examples:

make preset good_old_foo editable

```
  fw preset edit good_old_foo
```

save

This command can be applied on being_edited preset to make it saved. Optionally a new name can given given to the preset during this operation. Editable presets cannot be accidentally selected, and saved presets cannot be accidentally edited or removed. This is atomic operation – even if more than one actor can access EG at the same time they will receive error if they try to perform conflicting atomic actions. Atomic operations are: preset selection, creating new preset, saving or making preset edited, reading all presets for backup or restoring all presets from backup.

Synopsis:

```
fw preset save [-h] [-d DESTINATION] [name]
```

Detailed description of positional arguments:

```
  name                  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -d DESTINATION, --destination DESTINATION
                        Optional name under which currently being_edited
                        preset shall be saved, if not given current name of
                        preset being_edited will be used
```

Examples:

assuming only one preset is being edited currently save it under name `new_name`

```
fw preset save -d new_name
```

save preset which name starts with `my_` (there must be exactly one such preset in being edited state, but there may be other being edited presets whose names start differently

```
fw preset save 'my_*'
```

save the only being edited preset

```
fw preset save
```

select

This command can be applied on saved preset to apply its contents as current configuration of EG. Any previously selected preset will cease to be selected anymore. This operation may require confirmation by the user after it is applied to prevent accidental loss of communication between user and EG. If such confirmation is not received it will be rolled back. If such confirmation is required, then operation starts at the moment of execution and ends at the moment it is rolled back or commited. Selected preset cannot be made editable. This is atomic operation – even if more than one actor can access EG at the same time they will receive error if they try to perform conflicting atomic actions. Atomic operations are: preset selection, creating new preset, saving or making preset edited, reading all presets for backup or restoring all presets from backup.

Synopsis:

```
fw preset select [-h] [--unconditionally] [name]
```

Detailed description of positional arguments:

```
  name                  Optional shell glob pattern to limit names of presets to
                        print (defaults to '*')
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  --unconditionally     Prevent connectivity checking after executing this
                        command (without this option user may be asked to confirm
                        that after command execution he has not lost connection
                        to the device, without such confirmation command would be
                        rolled back.)
```

Examples:

select and apply configuration in preset `location2_config`

```
fw preset select location2_config
```

list

Lists existing presets (optionally limiting the result to presets matching glob pattern. Useful before executing commands to which concrete name is needed, or to check if preset is saved or being edited.

Synopsis:

```
fw preset list [-h] [name]
```

Detailed description of positional arguments:

```
name        Optional shell glob pattern to limit names returned (defaults to
            '*')
```

Detailed description of named arguments:

```
-h, --help  show this help message and exit
```

Examples:

list all presets

```
fw preset list
```

list presets which names start with an ‘a’ and end with an ‘x’

```
fw preset list 'a*x'
```

print

Prints contents of preset(s). For presets with multipart structure, if only one part is interesting output may be limited by giving the –part argument.

Synopsis:

```
fw preset print [-h] [-p PART] [name]
```

Detailed description of positional arguments:

```
name                  Optional shell glob pattern to limit names of presets
                      to print (defaults to '*')
```

Detailed description of named arguments:

```
-h, --help            show this help message and exit
-p PART, --part PART  Part (at least container of rules) of preset which
                      shall be printed. Whole preset will printed if not
                      given.
```

Examples:

print contents of part `foo/bar/baz` in all presets

```
fw preset print --part foo/bar/baz
```

print contents of whole `foobar` preset

```
fw preset print foobar
```

5.3.11 modify

Modifies contents of preset. Before actual modifaction can happen preset has to be in being_edited state (see `-E` in subcommands of this command and comands `fw preset edit` and `fw preset create`).

This command has following subcommands:

```
  copy                  copy a part of another preset to preset being_edited
  erase                 erase a part of preset being_edited
  set_policy            set a default treatment of packets if no other rule
                        matches
  common                modify common filtering rules (part
                        inet/filter/common/mgmtd)
  input                 modify filtering rules on packets targeted to EG (part
                        inet/filter/input/mgmtd)
  output                modify filtering rules on packets created by EG (part
                        inet/filter/output/mgmtd)
  forward               modify filtering rules on packets routed through EG
                        (part inet/filter /forward/mgmtd)
  ingress               Modify rules on packets routed through EG (part
                        netdev/ingress)
  create-chain-ingress
                        Create a chain with ingress hook (part
                        netdev/ingress).
  remove-chain-ingress
                        Remove a chain with ingress hook (part
                        netdev/ingress).
  nat_pre               modify nat rules applied before routing and filtering
                        (part ip/nat/prerouting/mgmtd)
  nat_post              modify nat rules applied after routing and filtering
                        (part ip/nat/postrouting/mgmtd)
  nat_n_on_n            allow creation of n:n static NAT with public and
                        private IP addressess
  masquerade            allow hiding private subnet or interface behind public
                        IP of another interface
  snat                  allow hiding private subnet or interface behind public
                        IP of another interface
  port_forward          allow keeping a service running on a private IP
                        available through public IP
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
```

Examples:

copy part `inet/filter/common` from preset `foo` to currently being edited preset (we did not provide `--name` option, hence only one preset is currently being edited)

```
fw modify copy -s foo -p inet/filter/common
```

first tries to mark saved preset `bar` as being edited, if it succeeds adds new masquerade rule named `masq_lan1` which will cause interface `lan1` to be treated as public one

```
fw modify masquerade --name bar -E add masq_lan1 --public_interface lan1
```

allows any incoming packets to port 22

```
fw modify input add allow_incoming_ssh -p ssh -v accept
```

disallows any outgoing packets not allowed by a specific rule — in connection with previous example ssh will not work, because outgoing traffic will be blocked

```
fw modify set_policy -p inet/filter/output drop
```

allows any outgoing packets belonging to (or related to) existing connections — this rectifies issue created by the drop policy from previous example because incoming packets will be allowed to port 22 (so connection will be created) and any packets related to existing connection will be allowed on output

```
fw modify output add allow_related --related -v accept
```

copy

Copies a part of another preset to preset being_edited. Useful when building new preset from parts of other existing presets by copying and erasing.

Synopsis:

```
fw modify copy [-h] [-n NAME] -s SOURCE [-p PART] [-E]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -s SOURCE, --source SOURCE
                        Name of preset which shall be source of data for
                        preset being_edited. Shell glob patterns may be used,
                        but exactly one name must match pattern.
  -p PART, --part PART  Part (at least container of rules) of preset which
                        shall be copied. Whole source preset will be copied if
                        not given.
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Examples:

copy part `inet/filter/common` from preset `foo` to currently being edited preset (we did not provide `--name` option, hence only one preset is currently being edited)

```
fw modify copy -s foo -p inet/filter/common
```

erase

Erases a part of preset being_edited. If the part is container for other parts erasing is equivalent to recreating erased part as copy from alllow_all preset. Useful when building new preset from parts of other existing presets by copying and erasing.

Synopsis:

```
fw modify erase [-h] [-n NAME] [-p PART] [-E]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -p PART, --part PART  Part of preset which shall be erased. Whole preset
                        will be affected if this parameter is not given.
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Examples:

marks save presed `foo` as being edited and removes part `inet/filter/common` from it

```
  fw modify erase -E -n foo -p inet/filter/common
```

set_policy

Synopsis:

```
fw modify set_policy [-h] [-n NAME] [-p PART] [-r] [-E] {accept,drop}
```

Detailed description of positional arguments:

```
  {accept,drop}         drop is definitive, accept means that later chains
                        will see the packet and may drop it later
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -p PART, --part PART  Part of preset which will be affected
  -r, --recursive       If given policy will be set recusively in all subparts
                        of part where applicable. If not given policy will be
                        set only on the part selected (and the part must be
                        having a policy
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
```

(continues on next page)

(continued from previous page)

```
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Sets policy which affects packets not matched by other specific rules. Option -r allows to set policy recursively in all subparts of part given by -p. Affected subparts list is returned by this command. Not matching any subpart is reported as error.

Examples

The following two commands will lead to dropping by default any externally generated packets, and accepting by default any packets orignated from the Edge Gateway.

```
fw modify set_policy -r -p inet/filter drop
fw modify set_policy -p inet/filter/output accept
```

System behavior

Note that drops from rules are immediate, and drop from policy happens only after all rules in a part have been checked. Note that accept from rule or policy in one part is not final, because packet may be revaluated but another part later. In general NAT prerouting rules are executed first, then filter rules are executed, finally NAT postrouting rules happen. If there is some more complicated preset, its ordering of rules shall be described in notes which accopany it.

common

Modifies single filtering rule common for packets incoming, outgoing and forwarded by EG (part inet/filter/common/mgmtd). The `mgmtd` indicates that rules are directly editable by the user. There might be other parts in factory prepared presets which can be copied with help of copy command, but which cannot be modified to avoid hard to detect inconsistencies in firewall rules.

Synopsis:

```
fw modify common [-h] [-n NAME] [-v {accept,drop}]
                        [-l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}]
                        [-p DESTINATION_PORT] [-P SOURCE_PORT] [-s SOURCE_IP]
                        [-S SOURCE_IP6] [-t DESTINATION_IP]
                        [-T DESTINATION_IP6] [-r] [-i INPUT_INTERFACE]
                        [-o OUTPUT_INTERFACE] [-M RAW_MATCH] [-A RAW_ACTION]
                        [-d NOTES] [-E]
                        {add,remove,edit,comment,uncomment} rule_name
```

Detailed description of positional arguments:

```
  {add,remove,edit,comment,uncomment}
                        What to do with a rule
  rule_name             Name of the rule to be affected
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
```

(continues on next page)

```
                        is being edited, so this argument is usally skipped.
-v {accept,drop}, --verdict {accept,drop}
                        drop or accept packet, if not given will be inverse of
                        policy at the time of rule addition, obviously if
                        there is no policy it cannot be skipped
-l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}, --protocol {ip,ip6,tcp,udp,sctp,icmp,icmpv6}
                        match only packets of given protocol
-p DESTINATION_PORT, --destination_port DESTINATION_PORT
                        match TCP and UDP packets with given destination port
                        (use --protocol if you want to match only TCP or only
                        UDP)
-P SOURCE_PORT, --source_port SOURCE_PORT
                        match TCP and UDP packets with given source port (use
                        --protocol if you want to match only TCp or only UDP)
-s SOURCE_IP, --source_ip SOURCE_IP
                        match only IPv4 packets with given source address
                        (accepts also: IP_ADDRESS/subnet)
-S SOURCE_IP6, --source_ip6 SOURCE_IP6
                        match only Ipv6 packets with given source address
                        (accepts also: IP_ADDRESS/subnet)
-t DESTINATION_IP, --destination_ip DESTINATION_IP
                        match only IPv4 packets with given destination address
                        (accepts also: IP_ADDRESS/subnet)
-T DESTINATION_IP6, --destination_ip6 DESTINATION_IP6
                        match only IPv6 packets with given destination address
                        (accepts also: IP_ADDRESS/subnet)
-r, --related           match packets of (or related to) previously accepted
                        connection --- note that to use --related you need
                        another rule which will accept the initial packet of a
                        connection, (for example you have a rule which accepts
                        all outgoing packets and a rule which accepts related
                        incoming packets)
-i INPUT_INTERFACE, --input_interface INPUT_INTERFACE
                        match packets which entered via given interface
-o OUTPUT_INTERFACE, --output_interface OUTPUT_INTERFACE
                        match packets which would exit via given interface
-M RAW_MATCH, --raw_match RAW_MATCH
                        value will be used in raw form for matching; Warning
                        --- in case of invalid contents error message will not
                        point to exact place of error
-A RAW_ACTION, --raw_action RAW_ACTION
                        value will be used in raw form as action to be
                        executed on matched packets; Warning --- in case of
                        invalid contents error message will not point to exact
                        place of error
-d NOTES, --description NOTES, --notes NOTES
                        additional notes about rule
-E, --make_edited       Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Examples:

add a new rule (with action not matching description!)

```
fw modify common add drop_smtp -l tcp -p 25 -v accept -d "This rule shall drop any SMTP packets"
```

correct previously created rule

```
fw modify common edit drop_smtp -l tcp -p 25 -v drop
```

input

Modifies single rule in part inet/filter/input/mgmtd. The `mgmtd` indicates that rules are directly editable by the user. There might be other parts in factory prepared presets which can be copied with help of copy command, but which cannot be modified to avoid hard to detect inconsistencies in firewall rules.

Synopsis:

```
fw modify input [-h] [-n NAME] [-v {accept,drop}]
                       [-l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}]
                       [-p DESTINATION_PORT] [-P SOURCE_PORT] [-s SOURCE_IP]
                       [-S SOURCE_IP6] [-t DESTINATION_IP]
                       [-T DESTINATION_IP6] [-r] [-i INPUT_INTERFACE]
                       [-o OUTPUT_INTERFACE] [-M RAW_MATCH] [-A RAW_ACTION]
                       [-d NOTES] [-E]
                       {add,remove,edit,comment,uncomment} rule_name
```

Detailed description of positional arguments:

```
  {add,remove,edit,comment,uncomment}
                        What to do with a rule
  rule_name             Name of the rule to be affected
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -v {accept,drop}, --verdict {accept,drop}
                        drop or accept packet, if not given will be inverse of
                        policy at the time of rule addition, obviously if
                        there is no policy it cannot be skipped
  -l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}, --protocol {ip,ip6,tcp,udp,sctp,icmp,icmpv6}
                        match only packets of given protocol
  -p DESTINATION_PORT, --destination_port DESTINATION_PORT
                        match TCP and UDP packets with given destination port
                        (use --protocol if you want to match only TCP or only
                        UDP)
  -P SOURCE_PORT, --source_port SOURCE_PORT
                        match TCP and UDP packets with given source port (use
                        --protocol if you want to match only TCp or only UDP)
  -s SOURCE_IP, --source_ip SOURCE_IP
                        match only IPv4 packets with given source address
                        (accepts also: IP_ADDRESS/subnet)
  -S SOURCE_IP6, --source_ip6 SOURCE_IP6
                        match only Ipv6 packets with given source address
                        (accepts also: IP_ADDRESS/subnet)
  -t DESTINATION_IP, --destination_ip DESTINATION_IP
                        match only IPv4 packets with given destination address
                        (accepts also: IP_ADDRESS/subnet)
  -T DESTINATION_IP6, --destination_ip6 DESTINATION_IP6
                        match only IPv6 packets with given destination address
                        (accepts also: IP_ADDRESS/subnet)
```

(continues on next page)

(continued from previous page)

```
  -r, --related         match packets of (or related to) previously accepted
                        connection --- note that to use --related you need
                        another rule which will accept the initial packet of a
                        connection, (for example you have a rule which accepts
                        all outgoing packets and a rule which accepts related
                        incoming packets)
  -i INPUT_INTERFACE, --input_interface INPUT_INTERFACE
                        match packets which entered via given interface
  -o OUTPUT_INTERFACE, --output_interface OUTPUT_INTERFACE
                        match packets which would exit via given interface
  -M RAW_MATCH, --raw_match RAW_MATCH
                        value will be used in raw form for matching; Warning
                        --- in case of invalid contents error message will not
                        point to exact place of error
  -A RAW_ACTION, --raw_action RAW_ACTION
                        value will be used in raw form as action to be
                        executed on matched packets; Warning --- in case of
                        invalid contents error message will not point to exact
                        place of error
  -d NOTES, --description NOTES, --notes NOTES
                        additional notes about rule
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Examples:

add a new rule (with action not matching description!)

```
  fw modify input add drop_smtp -l tcp -p 25 -v accept -d "This rule shall drop any SMTP packets"
```

correct previously created rule

```
  fw modify input edit drop_smtp -l tcp -p 25 -v drop
```

output

Modifies single rule on packets created by EG (part inet/filter/output/mgmtd). The `mgmtd` indicates that rules are directly editable by the user. There might be other parts in factory prepared presets which can be copied with help of copy command, but which cannot be modified to avoid hard to detect inconsistencies in firewall rules.

Synopsis:

```
fw modify output [-h] [-n NAME] [-v {accept,drop}]
                        [-l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}]
                        [-p DESTINATION_PORT] [-P SOURCE_PORT] [-s SOURCE_IP]
                        [-S SOURCE_IP6] [-t DESTINATION_IP]
                        [-T DESTINATION_IP6] [-r] [-i INPUT_INTERFACE]
                        [-o OUTPUT_INTERFACE] [-M RAW_MATCH] [-A RAW_ACTION]
                        [-d NOTES] [-E]
                        {add,remove,edit,comment,uncomment} rule_name
```

Detailed description of positional arguments:

```
  {add,remove,edit,comment,uncomment}
                        What to do with a rule
  rule_name             Name of the rule to be affected
```

Detailed description of named arguments:

```
-h, --help            show this help message and exit
-n NAME, --name NAME  Name of the preset being_edited which will be
                      affected. Shell glob patterns may be used but shall
                      match exactly one preset. If skipped this param
                      defaults to '*' and because normally only one preset
                      is being edited, so this argument is usally skipped.
-v {accept,drop}, --verdict {accept,drop}
                      drop or accept packet, if not given will be inverse of
                      policy at the time of rule addition, obviously if
                      there is no policy it cannot be skipped
-l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}, --protocol {ip,ip6,tcp,udp,sctp,icmp,icmpv6}
                      match only packets of given protocol
-p DESTINATION_PORT, --destination_port DESTINATION_PORT
                      match TCP and UDP packets with given destination port
                      (use --protocol if you want to match only TCP or only
                      UDP)
-P SOURCE_PORT, --source_port SOURCE_PORT
                      match TCP and UDP packets with given source port (use
                      --protocol if you want to match only TCp or only UDP)
-s SOURCE_IP, --source_ip SOURCE_IP
                      match only IPv4 packets with given source address
                      (accepts also: IP_ADDRESS/subnet)
-S SOURCE_IP6, --source_ip6 SOURCE_IP6
                      match only Ipv6 packets with given source address
                      (accepts also: IP_ADDRESS/subnet)
-t DESTINATION_IP, --destination_ip DESTINATION_IP
                      match only IPv4 packets with given destination address
                      (accepts also: IP_ADDRESS/subnet)
-T DESTINATION_IP6, --destination_ip6 DESTINATION_IP6
                      match only IPv6 packets with given destination address
                      (accepts also: IP_ADDRESS/subnet)
-r, --related         match packets of (or related to) previously accepted
                      connection --- note that to use --related you need
                      another rule which will accept the initial packet of a
                      connection, (for example you have a rule which accepts
                      all outgoing packets and a rule which accepts related
                      incoming packets)
-i INPUT_INTERFACE, --input_interface INPUT_INTERFACE
                      match packets which entered via given interface
-o OUTPUT_INTERFACE, --output_interface OUTPUT_INTERFACE
                      match packets which would exit via given interface
-M RAW_MATCH, --raw_match RAW_MATCH
                      value will be used in raw form for matching; Warning
                      --- in case of invalid contents error message will not
                      point to exact place of error
-A RAW_ACTION, --raw_action RAW_ACTION
                      value will be used in raw form as action to be
                      executed on matched packets; Warning --- in case of
                      invalid contents error message will not point to exact
                      place of error
-d NOTES, --description NOTES, --notes NOTES
                      additional notes about rule
-E, --make_edited     Mark saved preset as being edited one before
                      performing other actions, giving this option is the
                      same as if separate command `preset edit` was executed
                      by the user just before this command --- it may fail
                      if preset is already being edited, or if another user
                      tried to make preset edited at the same time
```

Examples:

add a new rule (with action not matching description!)

```
fw modify output add drop_smtp -l tcp -p 25 -v accept -d "This rule shall drop any SMTP packets"
```

correct previously created rule

```
fw modify output edit drop_smtp -l tcp -p 25 -v drop
```

forward

Modifies single rule on packets routed through EG (part inet/ filter/forward/mgmtd). The `mgmtd` indicates that rules are directly editable by the user. There might be other parts in factory prepared presets which can be copied with help of copy command, but which cannot be modified to avoid hard to detect inconsistencies in firewall rules.

Synopsis:

```
fw modify forward [-h] [-n NAME] [-v {accept,drop}]
                        [-l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}]
                        [-p DESTINATION_PORT] [-P SOURCE_PORT] [-s SOURCE_IP]
                        [-S SOURCE_IP6] [-t DESTINATION_IP]
                        [-T DESTINATION_IP6] [-r] [-i INPUT_INTERFACE]
                        [-o OUTPUT_INTERFACE] [-M RAW_MATCH] [-A RAW_ACTION]
                        [-d NOTES] [-E]
                        {add,remove,edit,comment,uncomment} rule_name
```

Detailed description of positional arguments:

```
  {add,remove,edit,comment,uncomment}
                        What to do with a rule
  rule_name             Name of the rule to be affected
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -v {accept,drop}, --verdict {accept,drop}
                        drop or accept packet, if not given will be inverse of
                        policy at the time of rule addition, obviously if
                        there is no policy it cannot be skipped
  -l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}, --protocol {ip,ip6,tcp,udp,sctp,icmp,icmpv6}
                        match only packets of given protocol
  -p DESTINATION_PORT, --destination_port DESTINATION_PORT
                        match TCP and UDP packets with given destination port
                        (use --protocol if you want to match only TCP or only
                        UDP)
  -P SOURCE_PORT, --source_port SOURCE_PORT
                        match TCP and UDP packets with given source port (use
                        --protocol if you want to match only TCp or only UDP)
  -s SOURCE_IP, --source_ip SOURCE_IP
                        match only IPv4 packets with given source address
                        (accepts also: IP_ADDRESS/subnet)
  -S SOURCE_IP6, --source_ip6 SOURCE_IP6
                        match only Ipv6 packets with given source address
                        (accepts also: IP_ADDRESS/subnet)
```

(continues on next page)

(continued from previous page)

```
  -t DESTINATION_IP, --destination_ip DESTINATION_IP
                        match only IPv4 packets with given destination address
                        (accepts also: IP_ADDRESS/subnet)
  -T DESTINATION_IP6, --destination_ip6 DESTINATION_IP6
                        match only IPv6 packets with given destination address
                        (accepts also: IP_ADDRESS/subnet)
  -r, --related         match packets of (or related to) previously accepted
                        connection --- note that to use --related you need
                        another rule which will accept the initial packet of a
                        connection, (for example you have a rule which accepts
                        all outgoing packets and a rule which accepts related
                        incoming packets)
  -i INPUT_INTERFACE, --input_interface INPUT_INTERFACE
                        match packets which entered via given interface
  -o OUTPUT_INTERFACE, --output_interface OUTPUT_INTERFACE
                        match packets which would exit via given interface
  -M RAW_MATCH, --raw_match RAW_MATCH
                        value will be used in raw form for matching; Warning
                        --- in case of invalid contents error message will not
                        point to exact place of error
  -A RAW_ACTION, --raw_action RAW_ACTION
                        value will be used in raw form as action to be
                        executed on matched packets; Warning --- in case of
                        invalid contents error message will not point to exact
                        place of error
  -d NOTES, --description NOTES, --notes NOTES
                        additional notes about rule
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Examples:

add a new rule (with action not matching description!)

```
  fw modify forward add drop_smtp -l tcp -p 25 -v accept -d "This rule shall drop any SMTP packets"
```

correct previously created rule

```
  fw modify forward edit drop_smtp -l tcp -p 25 -v drop
```

ingress

Modifies single rule on packets routed through EG (part netdev/ingress). The `mgmtd` indicates that rules are directly editable by the user. There might be other parts in factory prepared presets which can be copied with help of copy command, but which cannot be modified to avoid hard to detect inconsistencies in firewall rules.

Synopsis:

```
fw modify ingress [-h] [-n NAME] [-v {accept,drop}]
                        [-l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}]
                        [-p DESTINATION_PORT] [-P SOURCE_PORT] [-s SOURCE_IP]
                        [-S SOURCE_IP6] [-t DESTINATION_IP]
                        [-T DESTINATION_IP6] [-r] [-i INPUT_INTERFACE]
                        [-o OUTPUT_INTERFACE] [-M RAW_MATCH] [-A RAW_ACTION]
                        [-d NOTES] [-E]
```

(continues on next page)

(continued from previous page)

```
                        chain_name {add,remove,edit,comment,uncomment}
                        rule_name
```

Detailed description of positional arguments:

```
  chain_name            name of the chain to modify
  {add,remove,edit,comment,uncomment}
                        What to do with a rule
  rule_name             Name of the rule to be affected
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -v {accept,drop}, --verdict {accept,drop}
                        drop or accept packet, if not given will be inverse of
                        policy at the time of rule addition, obviously if
                        there is no policy it cannot be skipped
  -l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}, --protocol {ip,ip6,tcp,udp,sctp,icmp,icmpv6}
                        match only packets of given protocol
  -p DESTINATION_PORT, --destination_port DESTINATION_PORT
                        match TCP and UDP packets with given destination port
                        (use --protocol if you want to match only TCP or only
                        UDP)
  -P SOURCE_PORT, --source_port SOURCE_PORT
                        match TCP and UDP packets with given source port (use
                        --protocol if you want to match only TCp or only UDP)
  -s SOURCE_IP, --source_ip SOURCE_IP
                        match only IPv4 packets with given source address
                        (accepts also: IP_ADDRESS/subnet)
  -S SOURCE_IP6, --source_ip6 SOURCE_IP6
                        match only Ipv6 packets with given source address
                        (accepts also: IP_ADDRESS/subnet)
  -t DESTINATION_IP, --destination_ip DESTINATION_IP
                        match only IPv4 packets with given destination address
                        (accepts also: IP_ADDRESS/subnet)
  -T DESTINATION_IP6, --destination_ip6 DESTINATION_IP6
                        match only IPv6 packets with given destination address
                        (accepts also: IP_ADDRESS/subnet)
  -r, --related         match packets of (or related to) previously accepted
                        connection --- note that to use --related you need
                        another rule which will accept the initial packet of a
                        connection, (for example you have a rule which accepts
                        all outgoing packets and a rule which accepts related
                        incoming packets)
  -i INPUT_INTERFACE, --input_interface INPUT_INTERFACE
                        match packets which entered via given interface
  -o OUTPUT_INTERFACE, --output_interface OUTPUT_INTERFACE
                        match packets which would exit via given interface
  -M RAW_MATCH, --raw_match RAW_MATCH
                        value will be used in raw form for matching; Warning
                        --- in case of invalid contents error message will not
                        point to exact place of error
  -A RAW_ACTION, --raw_action RAW_ACTION
                        value will be used in raw form as action to be
```

(continues on next page)

(continued from previous page)

```
                        executed on matched packets; Warning --- in case of
                        invalid contents error message will not point to exact
                        place of error
  -d NOTES, --description NOTES, --notes NOTES
                        additional notes about rule
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Examples:

add a new rule (with action not matching description!)

```
  fw modify ingress add drop_smtp -l tcp -p 25 -v accept -d "This rule shall drop any SMTP packets"
```

correct previously created rule

```
  fw modify ingress edit drop_smtp -l tcp -p 25 -v drop
```

create-chain-ingress

Create a new chain with ingress hook for specified devices.

Synopsis:

```
fw modify create-chain-ingress [-h] [--chain-name CHAIN_NAME]
                                    [--device {} [{} ...]] [-n NAME] [-E]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  --chain-name CHAIN_NAME
                        Name of the chain (default is name of the
                        device/devices joined by '_').
  --device {} [{} ...]  LAN devices.
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

remove-chain-ingress

Synopsis:

```
fw modify remove-chain-ingress [-h] [-n NAME] [-E] chain_name
```

Detailed description of positional arguments:

```
  chain_name            Name of the chain.
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

nat_pre

Modifies single prerouting nat rule (part ip/nat/prerouting/mgmtd). Note that there are also specialized nat related commands (nat_n_on_n, masquerade, snat, port_forward). The `mgmtd` indicates that rules are directly editable by the user. There might be other parts in factory prepared presets which can be copied with help of copy command, but which cannot be modified to avoid hard to detect inconsistencies in firewall rules.

Synopsis:

```
fw modify nat_pre [-h] [-n NAME] [-v {accept,drop}]
                        [-l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}]
                        [-p DESTINATION_PORT] [-P SOURCE_PORT] [-s SOURCE_IP]
                        [-S SOURCE_IP6] [-t DESTINATION_IP]
                        [-T DESTINATION_IP6] [-r] [-i INPUT_INTERFACE]
                        [-o OUTPUT_INTERFACE] [-M RAW_MATCH] [-A RAW_ACTION]
                        [-d NOTES] [-E]
                        {add,remove,edit,comment,uncomment} rule_name
```

Detailed description of positional arguments:

```
  {add,remove,edit,comment,uncomment}
                        What to do with a rule
  rule_name             Name of the rule to be affected
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -v {accept,drop}, --verdict {accept,drop}
                        drop or accept packet, if not given will be inverse of
                        policy at the time of rule addition, obviously if
```

(continues on next page)

```
                        there is no policy it cannot be skipped
-l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}, --protocol {ip,ip6,tcp,udp,sctp,icmp,icmpv6}
                        match only packets of given protocol
-p DESTINATION_PORT, --destination_port DESTINATION_PORT
                        match TCP and UDP packets with given destination port
                        (use --protocol if you want to match only TCP or only
                        UDP)
-P SOURCE_PORT, --source_port SOURCE_PORT
                        match TCP and UDP packets with given source port (use
                        --protocol if you want to match only TCp or only UDP)
-s SOURCE_IP, --source_ip SOURCE_IP
                        match only IPv4 packets with given source address
                        (accepts also: IP_ADDRESS/subnet)
-S SOURCE_IP6, --source_ip6 SOURCE_IP6
                        match only Ipv6 packets with given source address
                        (accepts also: IP_ADDRESS/subnet)
-t DESTINATION_IP, --destination_ip DESTINATION_IP
                        match only IPv4 packets with given destination address
                        (accepts also: IP_ADDRESS/subnet)
-T DESTINATION_IP6, --destination_ip6 DESTINATION_IP6
                        match only IPv6 packets with given destination address
                        (accepts also: IP_ADDRESS/subnet)
-r, --related           match packets of (or related to) previously accepted
                        connection --- note that to use --related you need
                        another rule which will accept the initial packet of a
                        connection, (for example you have a rule which accepts
                        all outgoing packets and a rule which accepts related
                        incoming packets)
-i INPUT_INTERFACE, --input_interface INPUT_INTERFACE
                        match packets which entered via given interface
-o OUTPUT_INTERFACE, --output_interface OUTPUT_INTERFACE
                        match packets which would exit via given interface
-M RAW_MATCH, --raw_match RAW_MATCH
                        value will be used in raw form for matching; Warning
                        --- in case of invalid contents error message will not
                        point to exact place of error
-A RAW_ACTION, --raw_action RAW_ACTION
                        value will be used in raw form as action to be
                        executed on matched packets; Warning --- in case of
                        invalid contents error message will not point to exact
                        place of error
-d NOTES, --description NOTES, --notes NOTES
                        additional notes about rule
-E, --make_edited       Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Examples:

add a new rule (with action not matching description!)

```
fw modify nat_pre add drop_smtp -l tcp -p 25 -v accept -d "This rule shall drop any SMTP packets"
```

correct previously created rule

```
fw modify nat_pre edit drop_smtp -l tcp -p 25 -v drop
```

nat_post

Modifies single postrouting nat rule (part ip/nat/postrouting/mgmtd). Note that there are also specialized nat related commands (nat_n_on_n, masquerade, snat, port_forward). The `mgmtd` indicates that rules are directly editable by the user. There might be other parts in factory prepared presets which can be copied with help of copy command, but which cannot be modified to avoid hard to detect inconsistencies in firewall rules.

Synopsis:

```
fw modify nat_post [-h] [-n NAME] [-v {accept,drop}]
                         [-l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}]
                         [-p DESTINATION_PORT] [-P SOURCE_PORT]
                         [-s SOURCE_IP] [-S SOURCE_IP6] [-t DESTINATION_IP]
                         [-T DESTINATION_IP6] [-r] [-i INPUT_INTERFACE]
                         [-o OUTPUT_INTERFACE] [-M RAW_MATCH] [-A RAW_ACTION]
                         [-d NOTES] [-E]
                         {add,remove,edit,comment,uncomment} rule_name
```

Detailed description of positional arguments:

```
  {add,remove,edit,comment,uncomment}
                        What to do with a rule
  rule_name             Name of the rule to be affected
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -v {accept,drop}, --verdict {accept,drop}
                        drop or accept packet, if not given will be inverse of
                        policy at the time of rule addition, obviously if
                        there is no policy it cannot be skipped
  -l {ip,ip6,tcp,udp,sctp,icmp,icmpv6}, --protocol {ip,ip6,tcp,udp,sctp,icmp,icmpv6}
                        match only packets of given protocol
  -p DESTINATION_PORT, --destination_port DESTINATION_PORT
                        match TCP and UDP packets with given destination port
                        (use --protocol if you want to match only TCP or only
                        UDP)
  -P SOURCE_PORT, --source_port SOURCE_PORT
                        match TCP and UDP packets with given source port (use
                        --protocol if you want to match only TCp or only UDP)
  -s SOURCE_IP, --source_ip SOURCE_IP
                        match only IPv4 packets with given source address
                        (accepts also: IP_ADDRESS/subnet)
  -S SOURCE_IP6, --source_ip6 SOURCE_IP6
                        match only Ipv6 packets with given source address
                        (accepts also: IP_ADDRESS/subnet)
  -t DESTINATION_IP, --destination_ip DESTINATION_IP
                        match only IPv4 packets with given destination address
                        (accepts also: IP_ADDRESS/subnet)
  -T DESTINATION_IP6, --destination_ip6 DESTINATION_IP6
                        match only IPv6 packets with given destination address
                        (accepts also: IP_ADDRESS/subnet)
  -r, --related         match packets of (or related to) previously accepted
                        connection --- note that to use --related you need
                        another rule which will accept the initial packet of a
                        connection, (for example you have a rule which accepts
```

(continues on next page)

(continued from previous page)

```
                        all outgoing packets and a rule which accepts related
                        incoming packets)
  -i INPUT_INTERFACE, --input_interface INPUT_INTERFACE
                        match packets which entered via given interface
  -o OUTPUT_INTERFACE, --output_interface OUTPUT_INTERFACE
                        match packets which would exit via given interface
  -M RAW_MATCH, --raw_match RAW_MATCH
                        value will be used in raw form for matching; Warning
                        --- in case of invalid contents error message will not
                        point to exact place of error
  -A RAW_ACTION, --raw_action RAW_ACTION
                        value will be used in raw form as action to be
                        executed on matched packets; Warning --- in case of
                        invalid contents error message will not point to exact
                        place of error
  -d NOTES, --description NOTES, --notes NOTES
                        additional notes about rule
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Examples:

add a new rule (with action not matching description!)

```
  fw modify nat_post add drop_smtp -l tcp -p 25 -v accept -d "This rule shall drop any SMTP packets
↪"
```

correct previously created rule

```
  fw modify nat_post edit drop_smtp -l tcp -p 25 -v drop
```

nat_n_on_n

Allows creation of n:n static NAT with public and private IP addressess. Traffic to public addresses will be redirected to private addresses, and traffic from private addresses will appear as if it originated from public addresses

Synopsis:

```
fw modify nat_n_on_n [-h] [-n NAME]
                          [--public_interface PUBLIC_INTERFACE]
                          [--ip_public IP_PUBLIC] [--ip_private IP_PRIVATE]
                          [--mask MASK] [-d NOTES] [-E]
                          {add,remove,edit,comment,uncomment} rule_name
```

Detailed description of positional arguments:

```
  {add,remove,edit,comment,uncomment}
                        What to do with a rule
  rule_name             Name of the rule to be affected
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
```

(continues on next page)

(continued from previous page)

```
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  --public_interface PUBLIC_INTERFACE
                        optional interface name on which ip_public will be
                        visible
  --ip_public IP_PUBLIC
                        IP address --- traffic to ip_public will be redirected
                        to ip_private, and traffic from ip_private will appear
                        as if it originated from ip_public
  --ip_private IP_PRIVATE
                        IP address --- see ip_public option
  --mask MASK           optional netmask --- if not given 1:1 NAT will be
                        done, if given only netmask bits of ip_a and ip_b will
                        be translated (i.e. N:N nat will be performed)
  -d NOTES, --description NOTES, --notes NOTES
                        additional notes about rule
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Examples:

```
fw modify nat_n_on_n add from_lan1to2 --public_interface lan1 --ip_public 192.168.2.202 --ip_
↪private 192.168.1.102
```

masquerade

Allows hiding private subnet or interface behind public IP of another interface. To allow all needed traffic back firewall needs to track connections and protocols used by hidden machines. Automatically uses public IP of interface (which may be dynamic) and all connections are forgotten when interface goes down (difference from SNAT)

Synopsis:

```
fw modify masquerade [-h] [-n NAME]
                           [--public_interface PUBLIC_INTERFACE]
                           [--private_interface PRIVATE_INTERFACE]
                           [--ip_private IP_PRIVATE]
                           [--mask_private MASK_PRIVATE] [-d NOTES] [-E]
                           {add,remove,edit,comment,uncomment} rule_name
```

Detailed description of positional arguments:

```
  {add,remove,edit,comment,uncomment}
                        What to do with a rule
  rule_name             Name of the rule to be affected
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  --public_interface PUBLIC_INTERFACE
```

(continues on next page)

(continued from previous page)

```
                        interface with public IP
  --private_interface PRIVATE_INTERFACE
                        optional, interface behind which private subnets exist
  --ip_private IP_PRIVATE
                        optional, private IP address or subnet (if
                        mask_private is also given)
  --mask_private MASK_PRIVATE
                        optional netmask (may be given only if ip_private is
                        also given)
  -d NOTES, --description NOTES, --notes NOTES
                        additional notes about rule
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Examples:

simplest masquarade where lan2 has public dynamically assigned IP

```
  fw modify masquerade add masq_lan2 --public_interface lan2
```

snat

Allows hiding private subnet or interface behind public IP of another interface. To allow all needed traffic back firewall needs to track connections and protocols used by hidden machines. Public IP must be static, and connections may survive interface going down temporarily

Synopsis:

```
fw modify snat [-h] [-n NAME] [--ip_public IP_PUBLIC]
                      [--public_interface PUBLIC_INTERFACE]
                      [--private_interface PRIVATE_INTERFACE]
                      [--ip_private IP_PRIVATE] [--mask_private MASK_PRIVATE]
                      [-d NOTES] [-E]
                      {add,remove,edit,comment,uncomment} rule_name
```

Detailed description of positional arguments:

```
  {add,remove,edit,comment,uncomment}
                        What to do with a rule
  rule_name             Name of the rule to be affected
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  --ip_public IP_PUBLIC
                        Public IP address used as disguise for outgoing
                        traffic of private network
  --public_interface PUBLIC_INTERFACE
                        interface with public IP
  --private_interface PRIVATE_INTERFACE
```

(continues on next page)

(continued from previous page)

```
                        optional, interface behind which private subnets exist
  --ip_private IP_PRIVATE
                        optional, private IP address or subnet (if
                        mask_private is also given)
  --mask_private MASK_PRIVATE
                        optional netmask (may be given only if ip_private is
                        also given)
  -d NOTES, --description NOTES, --notes NOTES
                        additional notes about rule
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Examples:

simplest SNAT with public ip 43.21.35.17 on lan2

```
  fw modify snat add masq_lan2 --public_interface lan2 --public_ip 43.21.35.17
```

port_forward

Allows keeping a service running on a private IP available through public IP. ‘Only TCP and UDP services are supported.

Synopsis:

```
fw modify port_forward [-h] [-n NAME]
                            [--public_interface PUBLIC_INTERFACE]
                            [--ip_public IP_PUBLIC]
                            [--port_public PORT_PUBLIC]
                            [--ip_private IP_PRIVATE]
                            [--port_private PORT_PRIVATE]
                            [--protocol PROTOCOL] [-d NOTES] [-E]
                            {add,remove,edit,comment,uncomment} rule_name
```

Detailed description of positional arguments:

```
  {add,remove,edit,comment,uncomment}
                        What to do with a rule
  rule_name             Name of the rule to be affected
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  --public_interface PUBLIC_INTERFACE
                        optional, interface with public IP (to be used if
                        public IP is not known), if given only traffic
                        entering through this interface will be affected
  --ip_public IP_PUBLIC
                        optional, public IP address (shall always be used if
                        known, if not given public_interface becomes
```

(continues on next page)

(continued from previous page)

```
                        mandatory)
  --port_public PORT_PUBLIC
                        mandatory, port to which traffic will be forwarded
  --ip_private IP_PRIVATE
                        mandatory, private IP address to which traffic will be
                        forwarded
  --port_private PORT_PRIVATE
                        optional, allowed only if single protocol is being
                        used in rule, modifies port in forwarded packets
  --protocol PROTOCOL   optional, "tcp", or "udp", or "both", if not given
                        "tcp" will be assumed
  -d NOTES, --description NOTES, --notes NOTES
                        additional notes about rule
  -E, --make_edited     Mark saved preset as being edited one before
                        performing other actions, giving this option is the
                        same as if separate command `preset edit` was executed
                        by the user just before this command --- it may fail
                        if preset is already being edited, or if another user
                        tried to make preset edited at the same time
```

Examples:

```
fw modify port_forward add foo --public_interface lan1 --port_public 4321 --ip_private 192.168.2.
↪20 --port_private 1234 --protocol udp
```

5.3.12 print

Similar to `preset print` but intended to print only single currently edited preset. If there is only one preset being edited name of the preset can be skipped. If there is more than one preset being edited, name of the preset must be given and it shall match only one of them. –part argument allows to limit output, e.g. in case of huge preset.

Synopsis:

```
fw print [-h] [-n NAME] [-p PART]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name of the preset being_edited which will be
                        affected. Shell glob patterns may be used but shall
                        match exactly one preset. If skipped this param
                        defaults to '*' and because normally only one preset
                        is being edited, so this argument is usally skipped.
  -p PART, --part PART  Part (at least container of rules) of preset which
                        shall be printed. Whole preset will printed if not
                        given.
```

Examples:

print contents of the one and only currently being edited preset

```
  fw print
```

5.4 Container management

```
FORMAT controls the output.  Interpreted sequences are:
Usage:  docker [OPTIONS] COMMAND
A self-sufficient runtime for containers
Common Commands:
  run         Create and run a new container from an image
  exec        Execute a command in a running container
  ps          List containers
  build       Build an image from a Dockerfile
  pull        Download an image from a registry
  push        Upload an image to a registry
  images      List images
  login       Authenticate to a registry
  logout      Log out from a registry
  search      Search Docker Hub for images
  version     Show the Docker version information
  info        Display system-wide information
Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  checkpoint  Manage checkpoints
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  image       Manage images
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  plugin      Manage plugins
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes
Swarm Commands:
  config      Manage Swarm configs
  node        Manage Swarm nodes
  secret      Manage Swarm secrets
  service     Manage Swarm services
  stack       Manage Swarm stacks
  swarm       Manage Swarm
Commands:
  attach      Attach local standard input, output, and error streams to a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
```

(continues on next page)

(continued from previous page)

```
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  wait        Block until one or more containers stop, then print their exit codes
Global Options:
      --config string      Location of client config files (default
                           "/root/.docker")
  -c, --context string     Name of the context to use to connect to the
                           daemon (overrides DOCKER_HOST env var and
                           default context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket to connect to
  -l, --log-level string   Set the logging level ("debug", "info",
                           "warn", "error", "fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
                           "/root/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default
                           "/root/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default
                           "/root/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit
```

```
docker pull yoctobuild/yocto:first
docker images
docker ps
docker run -it -v yoctobuild/yocto:latest /bin/bash
```

5.5 Docker configuration

This command has following subcommands:

```
    apply               Restart docker service to apply changes docker DNS
                        configuration
    compose             Manage docker compose files to be started
                        automatically
    dns                 Add/remove docker DNS servers
    params              Configure docker daemon with parameters
                        {REQUIRES_APPLY}
    auth                Authenticate to private repositories
```

```
get_config option will generate docker-compose file for each currently running docker image that␣
↪was created by the user. This application will ignore docker created via Azure Edge.
```

```
docker-config get_config
```

```
docker-config load_compose --filename [docker-compose yml file]
```

5.5.1 apply

Some of the CLI commands allow to change the docker config in small increments. Restarting whole docker service after each of such small increments would take long time and could unnecesarily disrupt continuity of of work of the containers. Therefore documentation of such CLI commands notes, that you need to additionally trigger 'apply' command to restart docker service and apply all of your small incremental changes at convenient time.

Synopsis:

```
docker-config apply [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.5.2 compose

Containers defined by docker compose files can be automatically started after each reboot of Edge Gateway. Admin users can provide multiple compose files, which will be stored in centrally managed location and started by special user named 'composed'.

This command has following subcommands:

```
   recreate            force recreation of seleceted or all composed
                       containers
   get                 get contents of single docker-compose file
   status              saves artificial compose of all currently running
                       containers
   load                load docker-compose config file into system
   del                 remove docker-compose config from system
   set_config          replaces current compose files with new set
   get_config          get docker compose configuration and store it in a
                       file
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
```

Examples:

load contens of compose.yml in current directory under the name 'pump_regulator'

```
  docker-config compose load -f ./compose.yml -n pump_regulator
```

delete compose file added by above example

```
  docker-config compose del -n pump_regulator
```

get all compose files configured

```
  docker-config compose get_config
```

show yaml of currently running containers

```
  docker-config compose status
```

recreate

Example reason for recreation can be long lived container which holds some internal invalid state and is not working properly anymore. Another common case is need to change configuration values (e.g. proxy setting) embeded into container during its creation time — their subsequent change outside container will not be reflected unless recreation is forced.

Synopsis:

```
docker-config compose recreate [-h] [-n NAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  pattern for names to be recreated
```

get

Synopsis:

```
docker-config compose get [-h] -n NAME
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  name under which this compose file shall be stored
```

status

Synopsis:

```
docker-config compose status [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

This option can be used to save currently running docker containers to one docker-compose file which can be later used to start same set of containers manually (for example outside Edge Gateway). This option will only export docker containers spawned by the user. Docker containers used by Azure IoTEdge daemon will not be exported. Data will be saved to a file named `docker-compose.yml`.

Examples

```
docker-config compose status
```

System behavior

This option will not affect the system behavior.

load

Contents of docker compose yaml config will be loaded under name provided by –name option, and respective containers will be started immediately, and then after each reboot.

Synopsis:

```
docker-config compose load [-h] -f FILENAME -n NAME
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
                        path of the file with compose yaml config
  -n NAME, --name NAME  name under which this compose file shall be stored
```

Note that compose file will be stored in director of special user, which adds some limitations. For example relative paths will probably not work.

Examples

```
docker-config compose load --name flow_monitor --filename docker-compose.yml
```

System behavior

This option will change the list of currently running docker containers. In addition, docker-engine will create additional network interfaces, add firewall configuration to `iptables`.

del

Respective containers will be stopped and compose file removed.

Synopsis:

```
docker-config compose del [-h] -n NAME
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -n NAME, --name NAME  name of compose file to be removed
```

set_config

Restore docker compose files from file. This is a permanent change (and will remove previously present compose files (if any)) If containers are not running they will be started using new configuration. If you want to restart those which are running use `recreate` command.

Synopsis:

```
docker-config compose set_config [-h] [-f FILENAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
                        file with configuration data
```

get_config

All currently present docker compose files will be stored in to file named `compose_config.json`.

Synopsis:

```
docker-config compose get_config [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.5.3 dns

Add and remove DNS servers from your docker configuration.

This command has following subcommands:

```
    add                 Add DNS to current configuration (this command
                        requires executing 'apply' afterwards)
    del                 Delete DNS from current configuration (this command
                        requires executing 'apply' afterwards)
    set_config          Replace current docker DNS configuration
    get_config          Store current docker DNS configuration in file
                        dockerdns_config.json
    show                Show current docker DNS configuration
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
```

Examples:

add <IP_ADDR> as DNS for docker

```
  docker-config dns add <IP_ADDR>
```

remove <IP_ADDR> from list of DNS's of docker

```
  docker-config dns del <IP_ADDR>
```

replace current DNS list with one from file

```
  docker-config dns set_config -f <FILENAME>
```

store current configuration in file ./dockerdns_config.json

```
  docker-config dns get_config
```

display current nameservers from docker configuration

```
  docker-config dns show
```

add

Synopsis:

```
docker-config dns add [-h] ip_address
```

Detailed description of positional arguments:

```
  ip_address
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

del

Synopsis:

```
docker-config dns del [-h] ip_address
```

Detailed description of positional arguments:

```
  ip_address
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

set_config

Synopsis:

```
docker-config dns set_config [-h] [-f FILENAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
                        file with configuration data
```

get_config

Synopsis:

```
docker-config dns get_config [-h] [-f FILENAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
```

show

Synopsis:

```
docker-config dns show [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

5.5.4 params

Configuration of docker daemon

This command has following subcommands:

```
    show             Show all configured docker daemon parameters
    set_config       Set config for docker daemon.json
```

Detailed description of named arguments:

```
  -h, --help          show this help message and exit
  -M MTU, --mtu MTU   Set MTU on docker network interface
  -D, --debug         Enable docker debug
```

Examples:

sets given MTU for docker network interface (null for removal)

```
  docker-config params --mtu <MTU>
```

enable docker debug mode

```
  docker-config params --debug
```

show

Synopsis:

```
docker-config params show [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

set_config

Synopsis:

```
docker-config params set_config [-h] [-f FILENAME]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
                        file with configuration data
```

5.5.5 auth

Add or remove authentication to different repositories.

This command has following subcommands:

```
   add         Authenticate to private repositories
   remove      Remove url to private repositories
```

Detailed description of named arguments:

```
  -h, --help    show this help message and exit
```

add

Add authentication to different repositories.

Synopsis:

```
docker-config auth add [-h] [-u USER] [-p PASSWORD] [-U URL]
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -u USER, --user USER  name of the user
  -p PASSWORD, --password PASSWORD
                        password of the user
  -U URL, --url URL     url to the private repository (default is
                        'https://index.docker.io/v1/')
```

Examples:

```
docker-config auth add -u <user> -p <password> -U <url>
```

remove

Remove authentication to the repositories by providing the url.

Synopsis:

```
docker-config auth remove [-h] -U URL
```

Detailed description of named arguments:

```
  -h, --help         show this help message and exit
  -U URL, --url URL  url to the private repository to remove
```

Examples:

```
docker-config auth remove -U <url>
```

5.6 OVPN

This command has following subcommands:

```
    add                 Add new OpenVPN tunnel
    remove              Remove OpenVPN tunnel
    autostart           Disable/enable OpenVPN tunnel autostart
    connection          Disable/enable OpenVPN tunnel connection.
    get_config          Get configuration of OpenVPN tunnel and save it to
                        JSON file.
    set_config          Restore OpenVPN tunnel configuration from file
    show                Display OpenVPN tunnel configuration
    status              Display OpenVPN tunnel status
```

5.6.1 add

Synopsis:

```
ovpn add [-h] -f FILENAME -a {yes,no}
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
  -a {yes,no}, --autostart {yes,no}
```

This option can create a new tunnel configuration with an existing file. Function takes file with `.ovpn` extension for `-f/--filename` and `yes or no` for `-a/--autostart` as arguments. The autostart option defines whether the tunnel is automatically created during boot.

Examples

To create a new configuration based on the configuration file `tunnel.ovpn` with autostart enabled.

```
ovpn add -f tunnel.ovpn -a yes
```

```
ovpn add --filename tunnel.ovpn --autostart yes
```

System behaviour

This option will create a new tunnel configuration. If there is already a tunnel with the same name, CLI will ask to override the existing one.

5.6.2 remove

Synopsis:

```
ovpn remove [-h] -t {}
```

Detailed description of named arguments:

```
  -h, --help          show this help message and exit
  -t {}, --tunnel {}
```

This option can remove existing tunnel. The tunnel must be the same as the configuration file name without the `.ovpn` extension.

Examples

To remove configuration for tunnel `ovpnTunnel`.

```
ovpn remove -t ovpnTunnel
```

System behaviour

This option will stop selected tunnel and remove its configuration file from `/etc/openvpn`.

5.6.3 autostart

Synopsis:

```
ovpn autostart [-h] -t {} -a {enable,disable,status}
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -t {}, --tunnel {}    Tunnel config name
  -a {enable,disable,status}, --action {enable,disable,status}
                        To check current autostart option select status.
```

This option can be used to control the autostart behavior. One can also check the current status with `-a/--action` set to `status`. The tunnel must be the same as the configuration file name without the `.ovpn` extension.

Examples

To check current autostart for tunnel `ovpnTunnel`.

```
ovpn autostart -t ovpnTunnel -a status
```

To enable autostart for tunnel `ovpnTunnel`.

```
ovpn autostart -t ovpnTunnel -a enable
```

System behaviour

This option will change the behavior of the tunnel during the boot process.

5.6.4 connection

Synopsis:

```
ovpn connection [-h] -t {} -a {enable,disable,status}
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -t {}, --tunnel {}    Tunnel config name
  -a {enable,disable,status}, --action {enable,disable,status}
                        This will only enable/disable the tunnel for current
                        session. If you want to have this connection on boot
                        please check autostart option.
```

This option can be used to control the tunnel during system operation. One can also check the current status with `-a/--action` set to `status`. The tunnel must be the same as the configuration file name without the `.ovpn` extension.

Examples

To check current connection status for tunnel `ovpnTunnel`.

```
ovpn connection -t ovpnTunnel -a status
```

To enable tunnel `ovpnTunnel`.

```
ovpn connection -t ovpnTunnel -a enable
```

System behaviour

This option will change the behavior of the tunnel during current system operation.

5.6.5 get_config

Synopsis:

```
ovpn get_config [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

This option can be used to save current openvpn tunnels configuration to json file.

Examples

```
ovpn get_config
```

System behaviour

This option does not alter the system behavior or configuration.

5.6.6 set_config

Synopsis:

```
ovpn set_config [-h] -f FILENAME
```

Detailed description of named arguments:

```
  -h, --help            show this help message and exit
  -f FILENAME, --filename FILENAME
```

This option can be used to restore openvpn tunnels configuration from json file.

Examples

```
ovpn set_config --filename vpn.json
```

System behaviour

This option will change the current openvpn tunnels configuration. If there is already a configuration with the same name as in the configuration file, the existing one will be overwritten.

5.6.7 show

Synopsis:

```
ovpn show [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

This option can be used to print current openvpn configuration to console.

Examples

```
ovpn show
```

System behaviour

This option does not alter the system behavior or configuration.

5.6.8 status

Synopsis:

```
ovpn status [-h]
```

Detailed description of named arguments:

```
  -h, --help  show this help message and exit
```

This option can be used to print current openvpn tunnels status like IP address or device interface name.

Examples

```
ovpn status
```

System behaviour

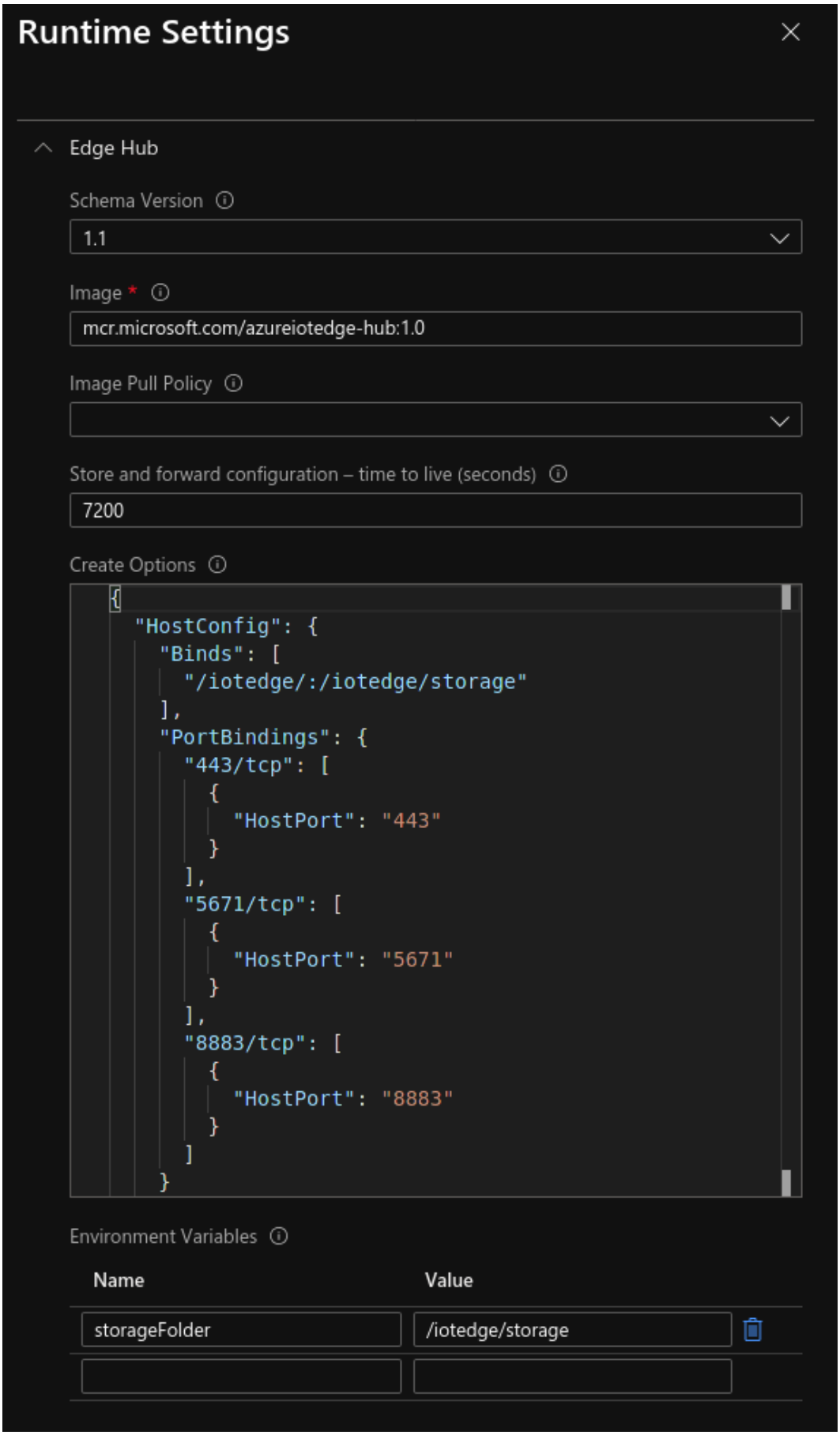
This option does not alter the system behavior or configuration.

5.7 Azure modules access to a device's local storage

One can use dedicated storage on the device in order to improve reliability, especially when operating offline.

5.8 Link module storage to device storage

Edge Gateway has dedicated storage place `/iotedge` that one can use to store module data. To enable module local storage one has to set environmental variable `storageFolder` in module configuration. That variable has to point to a catalog inside module's container. The module catalog has to be configured to use host directory `/iotedge` in order to keep data between the device reboot or system update.


Runtime Settings
Edge Hub
Schema Version
1.1
Image *
mcr.microsoft.com/azureiotedge-hub:1.0
Image Pull Policy
Store and forward configuration – time to live (seconds)
7200
Create Options
{
"HostConfig": {
"Binds": [
"/iotedge/:/iotedge/storage"
],
"PortBindings": {
"443/tcp": [
{
"HostPort": "443"
}
],
"5671/tcp": [
{
"HostPort": "5671"
}
],
"8883/tcp": [
{
"HostPort": "8883"
}
]
}
Environment Variables
Name
Value
storageFolder
/iotedge/storage

5.8.1 See more

For more information please check Microsoft Documentation available here: https://docs.microsoft.com/en-us/azure/iot-edge/how-to-access-host-storage-from-module?view=iotedge-2018-06#link-module-storage-to-device-storage

5.9 iotedge command

The iotedge tool is used to manage the IoT Edge runtime.

USAGE:

```
    iotedge [OPTIONS] <SUBCOMMAND>
```

OPTIONS:

```
    -h, --help           Print help information
    -H, --host <HOST>    Daemon socket to connect to [env: IOTEDGE_HOST=] [default:unix:///var/run/
↪iotedge/mgmt.sock]
    -V, --version        Print version information
```

SUBCOMMANDS:

```
    check              Check for common config and deployment issues
    check-list         List the checks that are run for 'iotedge check'
    help               Print this message or the help of the given subcommand(s)
    list               List modules
    logs               Fetch the logs of a module
    restart            Restart a module
    support-bundle     Bundles troubleshooting information
    version            Show the version information
```

5.9.1 iotedge check

Check for common config and deployment issues

USAGE:

```
    iotedge check [OPTIONS]
```

OPTIONS:

```
    -c, --config-file <FILE>
            Sets daemon configuration file [default: /etc/iotedge/config.yaml]

        --container-engine-config-file <FILE>
            Sets the path of the container engine configuration file [default:/etc/docker/daemon.
↪json]

        --diagnostics-image-name <IMAGE_NAME>
            Sets the name of the azureiotedge-diagnostics image. [default:mcr.microsoft.com/
↪azureiotedge-diagnostics:1.1.13]

        --dont-run <DONT_RUN>...
            Space-separated list of check IDs. The checks listed here will not be run. See
↪'iotedge check-list' for details of all checks.
            [possible values: certificates-quickstart, config-yaml-well-formed, connect-management-
↪uri, connection-string, container-connect-iothub-amqp, container-connect-iothub-https, container-
↪connect-iothub-mqtt,             container-default-connect-iothub-amqp, container-default-
↪connect-iothub-https, container-default-connect-iothub-mqtt, container-engine-dns, container-
↪engine-ipv6, container-engine-logrotate, container-engine-uri, container-local-time, edge-agent-
↪storage-mounted-from-host, edge-hub-storage-mounted-from-host, host-connect-dps-endpoint, host-
↪connect-iothub-amqp, host-connect-iothub-https, host-connect-iothub-mqtt, host-local-time,␣
↪hostname, identity-certificate-expiry, iotedged-version, windows-host-version]
```

(continues on next page)

(continued from previous page)

```
        --expected-iotedged-version <VERSION>
            Sets the expected version of the iotedged binary. Defaults to the value contained in
↪<http://aka.ms/latest-iotedge-stable>

    -h, --help
            Print help information

    -H, --host <HOST>
            Daemon socket to connect to [env: IOTEDGE_HOST=] [default:unix:///var/run/iotedge/mgmt.
↪sock]

        --iotedged <PATH_TO_IOTEDGED>
            Sets the path of the iotedged binary. [default: /usr/bin/iotedged]

        --iothub-hostname <IOTHUB_HOSTNAME>
            Sets the hostname of the Azure IoT Hub that this device would connect to. If using␣
↪manual provisioning, this does not need to be specified.

        --ntp-server <NTP_SERVER>
            Sets the NTP server to use when checking host local time. [default: pool.ntp.org:123]

    -o, --output <FORMAT>
            Output format. Note that JSON output contains some additional information like OS name,
↪ OS version, disk space, etc. [default: text] [possible values: json, text]

        --verbose
            Increases verbosity of output.

        --warnings-as-errors
            Treats warnings as errors. Thus 'iotedge check' will exit with non-zero code if it␣
↪encounters warnings.
```

5.9.2 iotedge check-list

List the checks that are run for 'iotedge check'

USAGE:

```
    iotedge check-list [OPTIONS]
```

OPTIONS:

```
    -h, --help           Print help information
    -H, --host <HOST>    Daemon socket to connect to [env: IOTEDGE_HOST=] [default:unix:///var/run/
↪iotedge/mgmt.sock]
```

5.9.3 iotedge list

List modules

USAGE:

```
    iotedge list [OPTIONS]
```

OPTIONS:

```
    -h, --help           Print help information
    -H, --host <HOST>    Daemon socket to connect to [env: IOTEDGE_HOST=] [default:unix:///var/run/
↪iotedge/mgmt.sock]
```

5.9.4 iotedge logs

Fetch the logs of a module

USAGE:

```
    iotedge logs [OPTIONS] <MODULE>
```

ARGS:

```
    <MODULE>    Sets the module identity to get logs
```

OPTIONS:

```
    -f, --follow
            Follow output log

    -h, --help
            Print help information

    -H, --host <HOST>
            Daemon socket to connect to [env: IOTEDGE_HOST=] [default:unix:///var/run/iotedge/mgmt.
↪sock]

        --since <DURATION or TIMESTAMP>
            Only return logs since this time, as a duration (1 day, 90 minutes, 2 days 3 hours 2␣
↪minutes), rfc3339 timestamp, or UNIX timestamp [default: "1 day"]

        --tail <NUM>
            Number of lines to show from the end of the log [default: all]

        --until <DURATION or TIMESTAMP>
            Only return logs up to this time, as a duration (1 day, 90 minutes, 2 days 3 hours 2␣
↪minutes), rfc3339 timestamp, or UNIX timestamp. For example, 0d would not truncate any logs,␣
↪while 2h would return logs up to 2 hours ago
```

5.9.5 iotedge restart

Restart a module

USAGE:

```
    iotedge restart [OPTIONS] <MODULE>
```

ARGS:

```
    <MODULE>    Sets the module identity to restart
```

OPTIONS:

```
    -h, --help           Print help information
    -H, --host <HOST>    Daemon socket to connect to [env: IOTEDGE_HOST=] [default:unix:///var/run/
↪iotedge/mgmt.sock]
```

5.9.6 iotedge support-bundle

Bundles troubleshooting information

USAGE:

```
    iotedge support-bundle [OPTIONS]
```

OPTIONS:

```
    -e, --include-edge-runtime-only
            Only include logs from Microsoft-owned Edge modules

    -h, --help
            Print help information

    -H, --host <HOST>
            Daemon socket to connect to [env: IOTEDGE_HOST=] [default:unix:///var/run/iotedge/mgmt.
↪sock]

        --iothub-hostname <IOTHUB_HOSTNAME>
            Sets the hostname of the Azure IoT Hub that this device would connect to. If␣
↪usingmanual provisioning, this does not need to be specified.

    -o, --output <FILENAME>
            Location to output file. Use - for stdout [default: support_bundle.zip]

    -q, --quiet
            Suppress status output

        --since <DURATION or TIMESTAMP>
            Only return logs since this time, as a duration (1d, 90m, 2h30m), rfc3339 timestamp,␣
↪or UNIX timestamp [default: "1 day"]

        --until <DURATION or TIMESTAMP>
            Only return logs up to this time, as a duration (1 day, 90 minutes, 2 days 3 hours 2␣
↪minutes), rfc3339 timestamp, or UNIX timestamp. For example, 0d would not truncate any logs,␣
↪while 2h would return logs up to 2 hours ago
```

5.9.7 iotedge version

Show the version information

USAGE:

```
    iotedge version [OPTIONS]
```

OPTIONS:

```
    -h, --help           Print help information
    -H, --host <HOST>    Daemon socket to connect to [env: IOTEDGE_HOST=] [default:unix:///var/run/
↪iotedge/mgmt.sock]
```

6 EdgeGateway WebUI Documentation

This document walks you through enabling, configuring, and using the EdgeGateway (eG-OS) WebUI to monitor and manage your device.

1. *Getting Started*
   - *Enabling the WebUI Service*
   - *Accessing the WebUI*
2. *Home Dashboard*
3. *Device Configuration*
4. *Network Settings*
5. *Cellular Settings*
6. *System Update*
7. *System Logs*
8. *Logging Out*

6.1 Getting Started

6.1.1 Enabling the WebUI Service

Use the CLI to enable and start the WebUI service:

```
device webgui enable
device webgui status
```

A successful status response looks like:

```
{
  "webgui": {
    "is_running": true,
    "is_enabled": true,
    "http_redirect": true,
    "subservice_status": "OK",
    "port": 48366
  }
}
```

Additional commands:

Command	Description
`device webgui disable`	Disable WebUI
`device webgui config --port <port>`	Change listening port
`device webgui redirect --enable`	Enable HTTP→HTTPS redirection
`device webgui redirect --disable`	Disable HTTP→HTTPS redirection