Donyx
Command Line Interface

Table of Contents

1. Command Line Interface (CLI) Access **4**
2. Basic Syntax and Navigation **4**
3. event **7**
4. firewall **7**
4.1. connections 7
4.2. defaults 8
4.3. filter 9
4.4. mangle 10
4.5. nat 11
4.6. raw 13
4.7. zone 14
5. ip **15**
5.1. arp 15
5.2. interface 15
5.3. route 17
5.3.1. list 17
5.3.2. rule 18
5.3.3. table 18
6. mobile **19**
6.1. apn 19
6.2. modem 20
6.3. sms 21
7. network **21**
7.1. bridge 21
7.2. device 23
7.3. ethernet 24
7.4. fdb 24
8. peripheral **24**
8.1. gpio 25
8.2. poe 25
8.3. protect 26
8.4. serial port 26
9. service **29**
9.1. client 29
9.2. dhcp 29
9.2.1. lease 29
9.2.2. server 30
9.3. dns 31
9.4. l2tp server 32
9.5. drm client 33
9.6. ntp 34
9.7. openvpn server 34
9.8. pinger 36
9.9. pptp server 37

9.10. snmp 38
9.11. vrrp 39
10. storage **40**
10.1. certificate 40
10.2. file 41
11. system **42**
11.1. access 42
11.1.1. radius 42
11.1.2. ssh 43
11.1.3. web 43
11.2. block dev 44
11.3. config 45
11.4. logging 45
11.5. management 46
11.6. package 47
11.7. user 47
12. tools **48**
13. tunnel **49**
13.1. eoip 49
13.2. gre 50
13.3. ipsec 51
13.3.1. association 51
13.3.2. connection 52
13.3.3. profile 53
13.3.4. proposal 53
13.3.5. status 54
13.4. l2tp 55
13.5. l2tp v3 56
13.6. openvpn 57
13.7. pptp 59
13.8. wireguard 59
13.8.1. interface 59
13.8.2. peer 60
14. wireless **61**
14.1. adapter 61
14.2. filter 62
14.3. network 62
15. Customer Support **65**

1. Command Line Interface (CLI) Access

The CLI is accessed via SSH. After successful authentication, the user is granted CLI access with privileges corresponding to their account.

Access Procedure:

1. Initiate an SSH connection to the router's IP address (default IP is on the device label)
2. Authenticate (enter password)
3. The device will automatically display the command line interface (cli)

```
user@desktop:~$ ssh admin@192.168.12.1
admin@192.168.12.1's password:
admin@Router[/]>
```

2. Basic Syntax and Navigation

TAB - show all sections

? - show all sections with hints

.. - go back one level

/ - return to root section

dashboard - display status information

export - export current device configuration. When called without arguments, the complete current configuration will be displayed in the console.

export ip - export configuration of a specific section

export [TAB] **to-file=** export to file

journal - system log. When called without arguments, the complete log data will be displayed in the console.

journal [TAB] **filter=** show lines containing specific information

journal [TAB] **last=** show last N lines

exit - close terminal

colorized - enable/disable color scheme

reload - refresh page

To enter a specific section, navigate sequentially through subsections:

```
admin@Router[/]>network
```

The response will be:

```
admin@Router[/network]>
```

You can also specify the full path to a section. The full path is specified sequentially, with spaces, without additional characters. Auto-completion using TAB key works.

For example:

```
 network bridge
```

The response will be:

```
admin@Router[/network bridge]>
```

To **view the contents of a section**, press **TAB** or **?**

```
admin@Router[/network bridge]>

  NAME       PORT   STP-VERSION
  bridge0    port1  none
             port2
             port3
             port4
             wifi1
  add
  apply
  clean
  export
  status
```

Adding or removing elements

To add or remove elements from a list, navigate to the corresponding section and write the name of the element to be added (for removal - with a minus sign)

For example:

View the list of ports on bridge0:

```
admin@Router[/network bridge bridge0]>port
```

The response will be:

```
port1,port2,port3,port4
```

Add port1 to the list of ports on bridge0:

```
admin@Router[/network bridge bridge0]>port port1
```

Remove port1 from the list of ports on bridge0:

```
admin@Router[/network bridge bridge0]>port -port1
```

3. event

The Event System section provides a flexible framework for automating router actions based on system or network state changes.

Unlike a simple if-then model, the system is built around modular, reusable components. This allows for the creation of sophisticated automation workflows.

COMMANDS

reorder	Change the position of an object

PROPERTIES

disabled **values:** true, false	Disable configuration
description	Script description for configuration notes
trigger	Event source subsystem
target	Source object of the event **conditions:** trigger = gpio trigger = gpi trigger = extra-gpi trigger = sms
value	The value of the object for comparison of the event **conditions:** trigger = gpio trigger = gpi trigger = sms
action	CLI command executed if the incoming event matches the configuration

4. firewall

4.1. connections

This section provides real-time monitoring and management of active network connections traversing the router. As a core component of the router's security framework, the Connection Tracker maintains stateful tracking of each connection, enabling informed decisions about traffic authorization and blocking. It offers visibility into which internal devices are establishing connections, the external IP addresses and ports they are communicating with, and the current state of those connections. This information enables the router to enforce firewall rules more effectively and enhance network security.

COMMANDS

flush	Flush connection tracking table

PROPERTIES

source	Source IP address for traffic filtering
destination	Destination IP address for routing decisions
protocol	IP protocol type selection
timeout	Connection timeout duration for session cleanup
tcp-state	TCP connection state tracking
mark	Packet mark for firewall rule matching

pkts	Transmitted/received packet counters
bytes	Transmitted/received byte counters

4.2. defaults

This section specifies the firewall's global default policies. These policies determine the final action for any network packet that does not match an explicit, user-defined rule. This establishes the router's default security posture, ensuring that all traffic is processed according to predefined rules. These base policies have the lowest priority and are only evaluated after all other specific rules have been processed.

The configuration is applied to the three main chains:

- **INPUT**: Controls traffic destined for the router's local processes, e.g., administrative access (HTTPS, SSH), ICMP responses, or VPN tunnel termination.
- **OUTPUT**: Controls traffic generated by the router itself, e.g., DNS queries, NTP client requests, or traffic initiated by connection monitoring tools.
- **FORWARD**: Controls traffic passing through the router between different network interfaces or zones, e.g., forwarding packets from the LAN zone to the WAN zone.

PROPERTIES

input **values:** accept, drop	Default input policy for incoming traffic
output **values:** accept, drop	Default output policy for outgoing traffic
forward **values:** accept, drop	Default forward policy for routed traffic
drop-invalid **values:** true, false	Drop invalid packets (malformed or suspicious)
syn-flood **values:** true, false	SYN flood attack protection
syn-flood-rate	TCP SYN packets per second threshold **conditions:** syn-flood = true
syn-flood-burst	TCP SYN burst limit for attack mitigation **conditions:** syn-flood = true
tcp-syncookies **values:** true, false	Enable TCP SYN cookies for flood protection
tcp-ecn **values:** true, false	Enable Explicit Congestion Notification (ECN)
tcp-window-scaling **values:** true, false	TCP window scaling for high-latency networks
use-conntrack **values:** true, false	Connection tracking (conntrack) for session monitoring
flow-offloading **values:** true, false	Flow offloading for hardware acceleration

4.3. filter

This section configures the primary rule set for packet filtering. These user-defined rules provide precise control over all traffic traversing or terminating at the router, and forms the core of the network security policy.

Each rule consists of specific match criteria — such as source/destination zones, IP addresses, protocols, and ports — and a designated target action (ACCEPT, DROP, or REJECT). The firewall evaluates these rules sequentially. The first rule that a packet matches determines its outcome, and processing for that packet stops.

Packets that do not match any rule in this list will be handled by the default policies configured in the *Firewall / Defaults* section. This mechanism allows for the creation of precise exceptions to the default security posture, enabling specific services and communications while maintaining an overall secure configuration.

COMMANDS

reset-counters	Reset traffic counters for this rule
reorder	Change the position of an object

PROPERTIES

disabled **values:** true, false	Disable firewall rule
chain **values:** input, forward, output	Firewall chain name for rule grouping
src **values:** /network device, /network ethernet, /wireless network, /ip interface, /mobile modem, /tunnel, /tunnel atunnel, /defaults server, /firewall zone	Traffic source zone or interface **conditions:** chain = output
src-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1:80, :80, :80,443,5000-5010	Source IP address range or host
dst **values:** /network device, /network ethernet, /wireless network, /ip interface, /mobile modem, /tunnel, /tunnel atunnel, /defaults server, /firewall zone	Traffic destination (interface or zone) **conditions:** chain = input
dst-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1:80, :80, :80,443,5000-5010	Destination IP address range or host
protocol **values:** dccp, ddp, egp, eigrp, encap, esp, etherip, ggp, gre, hmp, icmp, idpr-cmtp, idrp, igmp, igp, ip, ipcomp, ipencap, ipip, isis, iso-tp4, l2tp, ospf, pim, pup, rdp, rspf, rsvp, sctp, skip, st, tcp, udp, vmtp, vrrp, xns-idp, xtp, all	Protocol name matching for traffic classification

icmp-type **values:** address-mask-reply, host-redirect, pong, time-exceeded, address-mask-request, host-unknown, port-unreachable, timestamp-reply, any, host-unreachable, precedence-cutoff, timestamp-request, communication-prohibited, ip-header-bad, protocol-unreachable, TOS-host-redirect, destination-unreachable, network-prohibited, redirect, TOS-host-unreachable, echo-reply, network-redirect, required-option-missing, TOS-network-redirect, echo-request, network-unknown, router-advertisement, TOS-network-unreachable, fragmentation-needed, network-unreachable, router-solicitation, ttl-exceeded, host-precedence-violation, parameter-problem, source-quench, ttl-zero-during-reassembly, host-prohibited, ping, source-route-failed, ttl-zero-during-transit	match icmp packet type **conditions:** protocol = icmp
mark **pattern example:** 16, !453	Packet mark matching for firewall rule processing
action **values:** accept, reject, drop	Rule action
policy **values:** dir, pol, strict, reqid, spi, proto, mode, tunnel-src, tunnel-dst	IPsec policy matching criteria
extra **pattern example:** -m mac --mac-source FE:FF:FF:FF:FF:FF	iptables-compatible rule extensions

4.4. mangle

This section allows for the creation of rules to modify specific attributes of IP packets before they are processed by the routing or filtering engines. The primary function of the Mangle table is to classify and mark packets. These marks do not directly permit or block traffic but serve as internal identifiers for other router subsystems.

This functionality is essential for implementing advanced network features such as:

- **Policy-Based Routing (PBR)**: Directing marked packets through specific WAN interfaces or VPN tunnels, overriding the main routing table.
- **Quality of Service (QoS)**: Prioritizing or limiting the bandwidth for traffic flows based on their marks.

Additionally, Mangle rules can be used to directly alter IP header fields like DSCP (for QoS classification) or TTL (Time-to-Live).

COMMANDS

reset-counters	Reset traffic counters for this rule
reorder	Change the position of an object

PROPERTIES

disabled **values:** true, false	Disable firewall rule
chain **values:** prerouting, input, forward, output, postrouting	Firewall chain name for rule grouping
src **values:** /network device, /network ethernet, /wireless network, /ip interface, /mobile modem, /tunnel, /tunnel atunnel, /defaults server, /firewall zone	Traffic source zone or interface **conditions:** chain = output chain = postrouting

src-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1:80, :80, :80,443,5000-5010	Source IP address range or host
dst **values:** /network device, /network ethernet, /wireless network, /ip interface, /mobile modem, /tunnel, /tunnel atunnel, /defaults server, /firewall zone	Traffic destination (interface or zone) **conditions:** chain = input chain = prerouting
dst-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1:80, :80, :80,443,5000-5010	Destination IP address range or host
protocol **values:** dccp, ddp, egp, eigrp, encap, esp, etherip, ggp, gre, hmp, icmp, idpr-cmtp, idrp, igmp, igp, ip, ipcomp, ipencap, ipip, isis, iso-tp4, l2tp, ospf, pim, pup, rdp, rspf, rsvp, sctp, skip, st, tcp, udp, vmtp, vrrp, xns-idp, xtp, all	Protocol name matching for traffic classification
icmp-type **values:** address-mask-reply, host-redirect, pong, time-exceeded, address-mask-request, host-unknown, port-unreachable, timestamp-reply, any, host-unreachable, precedence-cutoff, timestamp-request, communication-prohibited, ip-header-bad, protocol-unreachable, TOS-host-redirect, destination-unreachable, network-prohibited, redirect, TOS-host-unreachable, echo-reply, network-redirect, required-option-missing, TOS-network-redirect, echo-request, network-unknown, router-advertisement, TOS-network-unreachable, fragmentation-needed, network-unreachable, router-solicitation, ttl-exceeded, host-precedence-violation, parameter-problem, source-quench, ttl-zero-during-reassembly, host-prohibited, ping, source-route-failed, ttl-zero-during-transit	ICMP type matching for diagnostic packet filtering **conditions:** protocol = icmp
mark **pattern example:** 16, !453	Packet mark matching for firewall rule processing
action **values:** accept, dscp, mark, tcpmss, ttl	Rule action **conditions:** chain = input
set-mark **minimum:** 0, **maximum:** 4294967295	Packet mark value for QoS/routing **conditions:** action = mark
set-dscp **values:** cs0, cs1, cs2, cs3, cs4, cs5, cs6, cs7, be, af11, af12, af13, af21, af22, af23, af31, af32, af33, af41, af42, af43, ef	Set DSCP value for traffic prioritization **conditions:** action = dscp
set-mss **values:** clamp-mss-to-pmtu	TCP MSS clamping for path MTU discovery **conditions:** action = tcpmss
set-ttl **pattern example:** 1, 255, +1, -1	Change packet time-to-live value **conditions:** action = ttl
policy **values:** dir, pol, strict, reqid, spi, proto, mode, tunnel-src, tunnel-dst	IPsec policy matching criteria
extra **pattern example:** -m mac --mac-source FE:FF:FF:FF:FF:FF	Custom iptables parameters

4.5. nat

This section is used to configure Network Address Translation (NAT) rules. These rules modify the source and/or destination IP addresses and ports of packets as they pass through the router. NAT is a core function for managing traffic between private and public networks.

The available actions are:

- **ACCEPT**: Explicitly disables NAT for matching traffic. This is used to create exceptions, for instance, to prevent address translation for traffic destined for a specific VPN tunnel or another private network.
- **SNAT**: Translates the source IP address to a specific, statically configured IP address. This action is typically used when the router's WAN interface has a fixed public IP.
- **Masquerade**: Dynamically translates the source IP address to the current IP address of the output interface. This is the standard and recommended action for interfaces with dynamic IP addresses, such as cellular (LTE) or DHCP connections.

COMMANDS

reset-counters	Reset traffic counters for this rule
reorder	Change the position of an object

PROPERTIES

disabled **values:** true, false	Disable firewall rule
chain **values:** prerouting, postrouting, output	Firewall chain name for rule grouping
src **values:** /network device, /network ethernet, /wireless network, /ip interface, /mobile modem, /tunnel, /tunnel atunnel, /defaults server, /firewall zone	Traffic source (zone or interface) **conditions:** chain = postrouting
src-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1:80, :80, :80,443,5000-5010	Source IP address range or host
dst **values:** /network device, /network ethernet, /wireless network, /ip interface, /mobile modem, /tunnel, /tunnel atunnel, /defaults server, /firewall zone	Traffic destination (interface or zone) **conditions:** chain = prerouting
dst-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1:80, :80, :80,443,5000-5010	Destination IP address range or host
protocol **values:** dccp, ddp, egp, eigrp, encap, esp, etherip, ggp, gre, hmp, icmp, idpr-cmtp, idrp, igmp, igp, ip, ipcomp, ipencap, ipip, isis, iso-tp4, l2tp, ospf, pim, pup, rdp, rspf, rsvp, sctp, skip, st, tcp, udp, vmtp, vrrp, xns-idp, xtp, all	Protocol name matching for traffic classification
icmp-type **values:** address-mask-reply, host-redirect, pong, time-exceeded, address-mask-request, host-unknown, port-unreachable, timestamp-reply, any, host-unreachable, precedence-cutoff, timestamp-request, communication-prohibited, ip-header-bad, protocol-unreachable, TOS-host-redirect, destination-unreachable, network-prohibited, redirect, TOS-host-unreachable, echo-reply, network-redirect, required-option-missing, TOS-network-redirect, echo-request, network-unknown, router-advertisement, TOS-network-unreachable, fragmentation-needed, network-unreachable, router-solicitation, ttl-exceeded, host-precedence-violation, parameter-problem, source-quench, ttl-zero-during-reassembly, host-prohibited, ping, source-route-failed, ttl-zero-during-transit	ICMP type matching for diagnostic packet filtering **conditions:** protocol = icmp
mark **pattern example:** 16, !453	Packet mark matching for firewall rule processing

action	Rule action **conditions:** chain = output chain = prerouting chain = postrouting
nat-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1:80, :80, :80,443,5000-5010	NAT translation for address masquerading **conditions:** chain = output && action = dnat chain = prerouting && action = dnat chain = postrouting && action = snat
redirect-port **pattern example:** 80, !80	Port forwarding to redirect traffic **conditions:** chain = output && action = redirect chain = prerouting && action = redirect
policy **values:** dir, pol, strict, reqid, spi, proto, mode, tunnel-src, tunnel-dst	IPsec policy matching criteria
extra **pattern example:** -m mac --mac-source FE:FF:FF:FF:FF:FF	Custom iptables parameters

4.6. raw

This section configures rules in the Raw table, the very first point of evaluation in the firewall's packet processing path. Rules here operate on "raw" packets before they are handled by the stateful connection tracking system (conntrack).

The primary purpose of this table is to identify specific traffic flows and apply theNOTRACKtarget, which instructs the firewall to completely bypass stateful inspection for those packets. This can be necessary for:

- **High-Performance Scenarios**: To reduce CPU and memory overhead on the router when handling extremely high rates of stateless traffic (e.g., from a local DNS resolver).
- **Protocol Compatibility**: To handle protocols that are inherently incompatible with stateful inspection or Network Address Translation (NAT).

Packets marked with NOTRACK are not processed by the stateful firewall engine. Consequently, they cannot be matched by state-based rules (e.g., RELATED, ESTABLISHED) and are incompatible with standard NAT functionality. This feature should be used with a clear understanding of its security implications.

COMMANDS

reset-counters	Reset traffic counters for this rule
reorder	Change the position of an object

PROPERTIES

disabled **values:** true, false	Disable firewall rule
chain **values:** output, prerouting	Firewall chain name for rule grouping
src **values:** /network device, /network ethernet, /wireless network, /ip interface, /mobile modem, /tunnel, /tunnel atunnel, /defaults server, /firewall zone	Traffic source zone or interface **conditions:** chain = output

src-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1:80, :80, :80,443,5000-5010	Source IP address range or host
dst **values:** /network device, /network ethernet, /wireless network, /ip interface, /mobile modem, /tunnel, /tunnel atunnel, /defaults server, /firewall zone	Traffic destination (interface or zone) **conditions:** chain = prerouting
dst-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1:80, :80, :80,443,5000-5010	Destination IP address range or host
protocol **values:** dccp, ddp, egp, eigrp, encap, esp, etherip, ggp, gre, hmp, icmp, idpr-cmtp, idrp, igmp, igp, ip, ipcomp, ipencap, ipip, isis, iso-tp4, l2tp, ospf, pim, pup, rdp, rspf, rsvp, sctp, skip, st, tcp, udp, vmtp, vrrp, xns-idp, xtp, all	Protocol name matching for traffic classification
icmp-type **values:** address-mask-reply, host-redirect, pong, time-exceeded, address-mask-request, host-unknown, port-unreachable, timestamp-reply, any, host-unreachable, precedence-cutoff, timestamp-request, communication-prohibited, ip-header-bad, protocol-unreachable, TOS-host-redirect, destination-unreachable, network-prohibited, redirect, TOS-host-unreachable, echo-reply, network-redirect, required-option-missing, TOS-network-redirect, echo-request, network-unknown, router-advertisement, TOS-network-unreachable, fragmentation-needed, network-unreachable, router-solicitation, ttl-exceeded, host-precedence-violation, parameter-problem, source-quench, ttl-zero-during-reassembly, host-prohibited, ping, source-route-failed, ttl-zero-during-transit	ICMP type matching for diagnostic packet filtering **conditions:** protocol = icmp
action **values:** accept, drop, notrack	Rule action
extra **pattern example:** -m mac --mac-source FE:FF:FF:FF:FF:FF	Custom iptables parameters

4.7. zone

This section is for creating and defining firewall zones. Zones are logical containers used to classify traffic sources and destinations, serving as the primary building blocks for security policies. A zone's membership is determined by its configured type:

- **Interface-based Zone**: Associates the zone with one or more network interfaces (e.g., eth0, br-lan, tun0). All traffic entering or leaving through a selected interface is considered part of this zone. This is the standard method for segmenting physical or logical network segments.
- **Address-based Zone**: Defines the zone by a list of IP addresses, CIDR subnets, or IP ranges. Traffic is considered part of this zone if its source or destination IP matches an entry in the list, regardless of the physical interface it traverses. This allows for creating policies that are independent of the network topology.

Once created, these zones can be used as source and destination targets to build a structured and effective security policy.

PROPERTIES

type **values:** interface, address	IP set type

entry	IP set member addresses **conditions:** type = interface type = address

5. ip

5.1. arp

This section provides an interface for viewing and managing the router's Address Resolution Protocol (ARP) table. ARP is a critical protocol used for resolving Layer 3 (IP) addresses into Layer 2 (MAC) addresses within a local network segment.
The page serves a dual purpose:

1. **ARP Cache Viewing**: It displays the current, dynamically populated ARP cache, showing all active IP-to-MAC address mappings that the router has learned automatically. For each entry, it provides stateful information, including the IP and MAC addresses, the associated network interface, and the entry's expiration time.

2. **Static ARP Entry Management**: The primary function of this section is to create static ARP entries. A static entry, or "ARP binding," creates a permanent, non-expiring mapping between an IP address and a specific MAC address. This is a key security and network management feature used to:

    - Enhance Security: Prevent ARP spoofing attacks, where a malicious device attempts to impersonate a legitimate network host by associating its MAC address with the victim's IP address.

    - Ensure Network Stability: Guarantee that critical devices (like servers, gateways, or IP cameras) are always reachable via a fixed IP-MAC pair, eliminating potential connectivity issues caused by ARP cache timeouts or incorrect dynamic resolutions.

Configuring a static entry ensures that the router will always trust the specified MAC address for the given IP address, ignoring any conflicting ARP announcements for that IP.

COMMANDS

flush	Clear Address Resolution Protocol (ARP) cache entries

PROPERTIES

ip-address **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1/255.255.255.0	IP address
macaddr **pattern example:** FE:FF:FF:FF:FF:FF	MAC address for hardware identification
interface **values:** /ip interface, /mobile modem, /tunnel eoip, /tunnel gre, /tunnel l2tp, /tunnel openvpn, /tunnel pptp, /tunnel wireguard interface, /tunnel l2tp-v3, /tunnel atunnel, /defaults server **pattern example:** bridge0	Network interface

5.2. interface

This section is dedicated to the configuration and management of the device's Layer 3 logical interfaces. Each interface is a logical entity that binds an IP address and associated L3 parameters to an underlying network device, forming the foundation for all routing, firewall, and service policies.

The configuration process involves setting common parameters applicable to all interfaces, such as administrative state, route metric, and DNS settings.

Additionally, protocol-specific options define the method of IP address configuration:

- **Static**: For the manual assignment of a fixed IPv4 or IPv6 address and subnet.
- **DHCP Client**: For dynamic IP address acquisition from an upstream DHCP server, with options for client identification.
- **PPPoE**: For establishing authenticated sessions using the Point-to-Point Protocol over Ethernet, including the configuration of authentication credentials and connection-specific options.

PROPERTIES

disabled **values:** true, false	Disable configuration
metric **minimum:** 0, **maximum:** 8388608	Route metric to determine path priority
ip-address	IP address assigned to the network interface **conditions:** type = static
defaultroute **values:** true, false	Add default route via interface for gateway traffic **conditions:** type != static
gateway **pattern example:** 192.168.1.1	Default gateway IP address for outbound traffic **conditions:** type = static
peer-dns **values:** true, false	Use assigned DNS servers for domain name resolution **conditions:** type = pppoe type = dhcp
pppoe-username	Point-to-Point Protocol over Ethernet (PPPoE) authentication username **conditions:** type = pppoe
pppoe-password	Point-to-Point Protocol over Ethernet (PPPoE) authentication password **conditions:** type = pppoe
pppoe-ac	PPPoE Access Concentrator (AC) name for service identification **conditions:** type = pppoe
pppoe-service	PPPoE client service configuration **conditions:** type = pppoe
pppoe-option **values:** lcp-echo-failure, lcp-echo-interval, lcp-max-configure, lcp-max-failure, lcp-max-terminate, lcp-restart, ipcp-accept-local, ipcp-accept-remote, ipcp-max-configure, ipcp-max-failure, ipcp-max-terminate, ipcp-restart, padi-attempts, padi-timeout, mru, mtu	PPP protocol advanced options **conditions:** type = pppoe
debug **values:** true, false	Verbose logging for detailed diagnostics **conditions:** type = pppoe

hostname	Dynamic Host Configuration Protocol (DHCP) client hostname override (optional) **conditions:** type = dhcp
dhcp-clientid	Dynamic Host Configuration Protocol (DHCP) client identifier (Option 61) **conditions:** type = dhcp
dhcp-vendorid	Dynamic Host Configuration Protocol (DHCP) vendor class identifier (Option 60) **conditions:** type = dhcp

5.3. route

5.3.1. list

This section displays the list of all user-configured static routes and serves as the primary interface for their management. Here, you can create new static entries, modify existing ones, or delete them as needed.

Each route configured in this list is a permanent instruction that is injected into the main kernel routing table. This allows for precise and persistent control over traffic paths, which is distinct from routes that are learned dynamically or created automatically for directly connected networks. The list provides a clear, consolidated view of all manually defined routing policies on the device.

PROPERTIES

disabled **values:** true, false	Disable configuration
dst-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1/255.255.255.0	Destination IP address for static route
gateway **pattern example:** 192.168.1.1	Next-hop gateway IP address for route forwarding **conditions:** type = unicast
metric **minimum:** 0, **maximum:** 8388608	Route metric value for path prioritization
table **values:** /ip route table, main	Routing table selection
src-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1/255.255.255.0	Source IP address for static route
type **values:** unicast, unreachable, prohibit, blackhole	Route type
interface **values:** /ip interface, /mobile modem, /tunnel eoip, /tunnel gre, /tunnel l2tp, /tunnel openvpn, /tunnel pptp, /tunnel wireguard interface, /tunnel l2tp-v3, /tunnel atunnel, /defaults server **pattern example:** bridge0	Egress interface for routed traffic **conditions:** type = unicast

5.3.2. rule

The IP / Rules section is the core of the Policy-Based Routing (PBR) engine. It allows for the creation of sophisticated traffic-steering policies by defining rules that match specific traffic attributes.

Unlike standard routing, which typically only considers the destination address, PBR rules can classify traffic based on a wider set of criteria, such as its source address, incoming interface, or firewall mark (fwmark). When a packet matches the conditions of a rule, it is directed to perform a route lookup in a specified routing table, overriding the default system-wide routing behavior.

This functionality is essential for implementing complex network scenarios like multi-WAN load balancing, routing traffic from specific VLANs through a VPN tunnel, or separating guest network traffic from internal corporate traffic. The rules are processed in order of their assigned priority, enabling precise control over the routing decision-making logic.

PROPERTIES

disabled **values:** true, false	Disable configuration
interface **values:** /ip interface, /mobile modem, /tunnel eoip, /tunnel gre, /tunnel l2tp, /tunnel openvpn, /tunnel pptp, /tunnel wireguard interface, /tunnel l2tp-v3, /tunnel atunnel, /defaults server, any **pattern example:** bridge0	Ingress interface matching for packet filtering
src-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1/255.255.255.0	Source IP address range or host matching
dst-addr **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1/255.255.255.0	Destination IP address range or host matching
mark **minimum:** 0, **maximum:** 4294967295	Connection tracking (conntrack) mark matching
table **values:** /ip route table	Routing table lookup for matched packets
action **values:** prohibit, unreachable, blackhole, throw	The type of action
priority **minimum:** 1, **maximum:** 9999999	Rule priority (lower values processed first)

5.3.3. table

This section is dedicated to the definition and management of custom routing tables. These tables are a core component of advanced routing scenarios, particularly Policy-Based Routing (PBR), which allows for different routing decisions to be made based on traffic characteristics.

While the system operates with several default tables (main, local, etc.), this interface allows administrators to declare additional, named routing tables. Each new table serves as an independent namespace for routing entries, enabling the creation of distinct routing policies for different types of traffic.

The population of these tables with static routes is performed in the **IP/Route/List** section, while the logic for directing specific traffic to be looked up in these custom tables is configured in the **IP/Rules**

section.

PROPERTIES

number **minimum:** 16384, **maximum:** 99999	Routing table numeric identifier

6. mobile

6.1. apn

This section is designed for the advanced configuration of mobile data connections, primarily for use with Mobile Virtual Network Operators (MVNOs).

MVNOs often utilize the infrastructure of multiple underlying physical carriers. To connect to a specific carrier's network, the MVNO may require a unique combination of an Access Point Name (APN) and a mobile network code (MCC/MNC). This section allows you to create predefined profiles, each containing a complete set of connection parameters (APN, MCCMNC, authentication method, etc.). The router can then automatically apply the correct profile when it detects the corresponding network, ensuring a successful connection.

For most major, non-virtual mobile operators, manual APN configuration is not required, as the modem negotiates these settings automatically with the network. Creating custom profiles is primarily necessary for ensuring reliable connectivity on MVNO SIMs or for manually forcing the device to use a specific network configuration.

PROPERTIES

disabled **values:** true, false	Disable configuration
mccmnc **pattern example:** 46000	PLMN code (MCC+MNC)
apn **pattern example:** bridge0, vpn-client1, user_1@example.com	APN or VPDN name
username **maxLength:** 39 **pattern example:** bridge0, vpn-client1, user_1@example.com	Client authentication username
password	Client authentication password
auth **values:** any, pap, chap	Authentication protocol
mode **values:** auto, /defaults modem_modes	Data transmission technology
roaming **values:** allowed, discard, only	Roaming network policy
slot **values:** any, /mobile slot	Assign profile to SIM card slot
pincode	SIM card PIN code **conditions:** slot != any

6.2. modem

This section is the primary control center for configuring the router's cellular modem and its connection to the internet. It provides a comprehensive set of tools to manage everything from basic connectivity to advanced failover and network selection policies.

The settings here allow you to:

- **Configure Routing Parameters:** Define how the cellular connection integrates with the router's routing table. You can control its preference using the Metric field and designate it as the Default Route.
- **Manage SIM Card Failover:** For models equipped with a modem that supports dual SIM functionality, this section provides robust tools for high availability. You can define a Primary SIM and a backup, and configure the Return to Primary interval to automatically switch back to the preferred SIM card once it becomes available again.
- **Fine-Tune Connection Reliability:** You can set a Connect Timeout to ensure the modem power-cycles and re-initiates a connection if it fails to register on the network within a specified period.
- **Control Network Registration:** For advanced scenarios, you can manually select Specific Bands to force the modem to use only certain frequencies, or use Force MCCMNC to lock the device to a specific operator's network. This is particularly useful for private LTE networks or for improving connection stability in areas with overlapping coverage.

Please note that the availability of certain features, such as multi-SIM management (Primary SIM, Return to Primary) and Specific Bands selection, is dependent on the specific router model. The page also displays key real-time status information about the active modem connection.

COMMANDS

reset	Restarting the modem using a power reset

PROPERTIES

disabled **values:** true, false	Disable cellular modem
modem-sim-slot **values:** $_available-sim-slots	Available SIM card slots for modem
modem-primary-sim **values:** $modem-sim-slot	Primary SIM card slot preference
modem-primary-sim-timeout	The time (sec) after which the modem will switch to the primary SIM card
modem-connect-timeout **minimum:** 300	SIM connection attempt timeout (sec)
modem-band **values:** $_available-bands	Preferred modem frequency bands
protocol **values:** auto, ppp	Modem dial-up protocol
metric **minimum:** 101, **maximum:** 900	Modem interface metric

defaultroute **values:** true, false	Add default route via interface for gateway traffic
peerdns **values:** true, false	Use assigned DNS servers for domain name resolution
mtu **minimum:** 400, **maximum:** 1500	Interface MTU to configure maximum transmission unit size

6.3. sms

This section provides an interface for sending and receiving Short Message Service (SMS) messages directly through the router's cellular module.

Reading SMS

The interface displays a table of the most recently received SMS messages. This acts as a rotating log, storing up to the last 10 messages. When a new SMS is received and the log is full, the oldest entry is automatically discarded to accommodate the new one.

Sending SMS

The sending utility allows you to compose and transmit SMS messages to any valid phone number. For a message to be sent successfully, the following conditions must be met:

- The installed SIM card must have an active SMS service plan and a sufficient account balance.
- The router must be successfully registered on the operator's network (i.e., be within a coverage area).

COMMANDS

send-sms	Send SMS to phone number phone-number - SMS recipient's phone number (required) message - SMS message text (required) modem - Send SMS via specific modem

PROPERTIES

from	SMS sender name/number
text	SMS message text
sent	SMS sending timestamp
received	SMS receipt timestamp

7. network

7.1. bridge

The Network / Bridge section is used to create and manage software-defined Layer 2 bridges. A bridge effectively functions as a virtual switch, grouping multiple physical or logical network interfaces (e.g., Ethernet ports, VLAN sub-interfaces) into a single, unified broadcast domain.

Beyond basic interface aggregation, this section provides a suite of advanced Layer 2 management features:

- **Spanning Tree Protocol (STP)**: Full configuration of STP/RSTP to prevent network loops in redundant physical topologies. This includes granular control over bridge priority, port states, and protocol timing to ensure a loop-free and stable network.
- **IGMP Snooping**: Capabilities to optimize the delivery of multicast traffic. When enabled, the bridge intelligently inspects IGMP communication and forwards multicast streams only to the ports that have explicitly requested them, significantly reducing unnecessary network traffic and improving efficiency.
- **Performance Tuning**: Control over low-level parameters such as MTU and transmit queue length to optimize performance for specific use cases.

PROPERTIES

disabled **values:** true, false	Disable configuration
macaddr **pattern example:** FE:FF:FF:FF:FF:FF	MAC address for hardware identification
tx-queue-len	Maximum interface queue packet capacity
port **values:** /network device, /network ethernet, /wireless network, /tunnel	Bridge port members
vlan-default-pvid **minimum:** 1, **maximum:** 4095	Native Virtual LAN (VLAN) ID for untagged frames **conditions:** vlan-filtering = true
igmp-snooping **values:** true, false	Internet Group Management Protocol (IGMP) snooping for multicast filtering
igmp-mc-querier **values:** true, false	Multicast router suppression to avoid duplicate streams **conditions:** igmp-snooping = true
igmp-query-interval	Internet Group Management Protocol (IGMP) query interval (sec) **conditions:** igmp-snooping = true
igmp-query-resp-interval	Internet Group Management Protocol (IGMP) host response timeout **conditions:** igmp-snooping = true
igmp-hash-max	Internet Group Management Protocol (IGMP) group table hash size **conditions:** igmp-snooping = true
igmp-robustness	Internet Group Management Protocol (IGMP) max unanswered queries **conditions:** igmp-snooping = true
stp-version **values:** none, rstp, stp	Spanning Tree Protocol (STP) version selection
stp-bpdu-guard **values:** $port	Enable Bridge Protocol Data Unit (BPDU) guard to disable port on unauthorized BPDU reception **conditions:** stp-version != none
stp-treeprio **minimum:** 1, **maximum:** 15	Spanning Tree Protocol (STP) bridge priority value **conditions:** stp-version != none

stp-maxage **minimum:** 6, **maximum:** 40	Spanning Tree Protocol (STP) max age timer for root bridge connectivity loss **conditions:** stp-version != none
stp-fdelay **minimum:** 6, **maximum:** 40	Spanning Tree Protocol (STP) port state transition delay **conditions:** stp-version != none
stp-maxhops **minimum:** 6, **maximum:** 40	Spanning Tree Protocol (STP) maximum hop count to restrict network diameter **conditions:** stp-version != none
stp-hello **minimum:** 1, **maximum:** 10	Spanning Tree Protocol (STP) hello interval for root bridge BPDU transmissions **conditions:** stp-version != none
stp-ageing **minimum:** 10, **maximum:** 1000000	Forwarding Database (FDB) MAC address aging timeout **conditions:** stp-version != none
stp-txholdcount **minimum:** 1, **maximum:** 10	Bridge Protocol Data Unit (BPDU) transmission rate limit **conditions:** stp-version != none

7.2. device

The Network / Devices section is responsible for the low-level management of Layer 2 network device objects. Unlike the physical hardware ports (e.g., eth0, eth1), the devices managed here are often virtual or logical constructs that enable advanced network configurations. This is the foundational layer where you create the building blocks for your network topology before they are used in higher-level configurations.

Once a logical device is defined here (with its own MAC address, MTU, etc.), it can be added to a bridge (in the Network / Bridge section) for L2 switching or configured with an IP address (in the IP / Interfaces section) to operate at Layer 3. This section provides direct control over the L2 entities that underpin the entire network stack.

PROPERTIES

disabled **values:** true, false	Disable configuration
type **values:** vlan	Device type
macaddr **pattern example:** FE:FF:FF:FF:FF:FF	Hardware (MAC) address of the device **conditions:** type != vlan
mtu **minimum:** 70, **maximum:** 65535	Interface MTU to configure maximum transmission unit size
tx-queue-len	Maximum interface queue packet capacity
device **values:** /network ethernet, /tunnel	Create a new device on top of an another device
vid **minimum:** 6, **maximum:** 4094	Virtual LAN (VLAN) tag identifier (802.1Q)

7.3. ethernet

The Network / Ethernet section provides direct control over the physical Layer 1 and data-link Layer 2 properties of the router's hardware Ethernet ports. These settings are fundamental as they define the physical operating parameters of a port before it is used in any higher-level logical configurations, such as being added to a bridge or assigned an IP interface.

This interface allows for fine-tuning port behavior to ensure compatibility with connected devices and to optimize network performance. Key adjustments include manually setting the **link speed and duplex mode** to resolve auto-negotiation issues, modifying the **MTU** to accommodate specific network requirements, and enabling **promiscuous mode** for advanced network traffic analysis and troubleshooting.

PROPERTIES

disabled **values:** true, false	Disable configuration
macaddr **pattern example:** FE:FF:FF:FF:FF:FF	Ethernet port MAC address
mtu **minimum:** 70, **maximum:** 1500	Ethernet interface Maximum Transmission Unit (MTU) size (byte)
promisc **values:** true, false	Enable promiscuous mode for unrestricted packet capture
speed **values:** /defaults ether_port_speed	Ethernet link speed negotiation
duplex **values:** half, full	Ethernet duplex mode **conditions:** speed = auto

7.4. fdb

The Network / Forwarding Database (FDB) section displays the bridge's dynamically learned MAC address table. This table is the core of Layer 2 switching, as it maps the MAC addresses of connected devices to the specific bridge ports (physical or logical) where they are located.

The table is populated automatically by inspecting the source MAC address of incoming frames. By using the FDB, the bridge sends unicast traffic directly to the correct destination port instead of flooding it to all ports. This is critical for network performance and security.

This read-only view is an essential diagnostic tool for troubleshooting Layer 2 connectivity, verifying network topology, and identifying which port a specific device is connected to.

PROPERTIES

mac	MAC address of networked device
device	Bridge port associated with MAC address
age	Forwarding Database (FDB) entry age timestamp
type	Device type (physical or virtual)

8. peripheral

8.1. gpio

This section provides an interface for configuring the router's General-Purpose Input/Output (GPIO) pins. GPIOs allow the router to interact with external circuits and devices, either by detecting state changes on input pins or by controlling external components via output pins.

The availability, quantity, and physical characteristics of the GPIO pins are specific to the router model. For detailed specifications, please consult the official user manual or the manufacturer's website.

- Voltage must only be applied to a GPIO input after the router is fully powered on.
- The voltage applied to any GPIO input must never exceed the router's main power supply voltage.
- Failure to comply with these requirements will result in permanent damage to the router and will void the manufacturer's warranty.

If a 10 kΩ resistor is not connected to the GPIO, the voltage applied to the input must not differ from the router's supply voltage. If a 10 kΩ resistor is installed, a difference between the router's supply voltage and the GPIO input voltage is permissible.

PROPERTIES

direction **values:** in, out	Port operating mode **conditions:** .type = gpi .type = gpo
value **values:** high, low	Voltage level on the port **conditions:** direction = in
trigger **values:** none, rising, falling, both	Edge-triggered event generation **conditions:** direction = in
debounce	Contact debounce interval (msec) **conditions:** direction = in

8.2. poe

This section provides control and monitoring for the Power over Ethernet (PoE) functionality, **available only on specific router models equipped with PoE-capable hardware.**

The router features a "PoE Defender" system, which not only supplies power to connected devices but also actively monitors each port for electrical faults (such as short circuits or overcurrent) and attempts to automatically recover from them.

Port Status and Operation

You can monitor the real-time status of each PoE-capable port:

- **Disabled:** The port is administratively off, and no power is being supplied.
- **Enabled:** The port is active and successfully delivering power to a connected device.
- **Error:** A fault has been detected, and the system has automatically cut power to the port for safety.

Fault Handling and Auto-Recovery

When a fault triggers the protection mechanism, the "PoE Defender" initiates an automatic recovery sequence:

1. The first recovery attempt is made **2 seconds** after the fault is detected.
2. If the first attempt fails, a second attempt is made **5 seconds** later.

If both attempts are unsuccessful, the port remains in the *Error* state to prevent damage to the router or the connected device, requiring manual intervention.

Manual Intervention

To manage a port in an *Error* state or to configure a port:

- **To check the fault:** Click on the specific port to view a description of the last recorded error and its timestamp.
- **To manually re-apply power:** After resolving the physical issue, click the **Apply** button to attempt to re-enable the port.
- **To enable/disable a port:** Select the desired port and toggle the **Disabled** checkbox in its settings.

PROPERTIES

disabled **values:** true, false	Disable PoE on port at system startup

8.3. protect

This section manages the **protectd** service, that provides active electrical protection for the General-Purpose I/O (GPIO) pins.

The service monitors the GPIOs for fault conditions such as short circuits or overcurrent. If a fault is detected, **Protectd** automatically disables the affected pin to prevent damage to the router's internal circuitry. The pin will remain in a disabled fault state until the issue is resolved and the service is manually restarted from this page.

The restart action should only be performed after the external physical fault (e.g., the short circuit in the connected wiring) has been rectified.

COMMANDS

restart	Restart GPIO protection

8.4. serial port

This section configures the router's physical RS232 and RS485 ports, transforming the device into a versatile Serial-to-IP gateway. This functionality allows you to encapsulate serial data into IP packets and transport it over a network, enabling remote access and integration of legacy serial equipment into modern IP-based systems.

The core of the configuration is the operational **Mode**, which defines how the router handles the serial data:

- **Server Mode:** The router listens on a specified TCP port for incoming connections. A remote client connects to the router to access the attached serial device.

- **Client Mode:** The router actively initiates a TCP connection to a remote server, forwarding any data received from the local serial port.
- **Modbus TCP Mode:** The router functions as a Modbus gateway, converting Modbus TCP requests from the network into Modbus RTU/ASCII requests for the connected serial slave device.
- **NTRIP Client Mode:** A specialized mode for connecting to an NTRIP Caster to receive GNSS correction data and forward it to an attached serial device (like a high-precision GPS receiver).

PROPERTIES

disabled **values:** true, false	Disable configuration
mode **values:** client, server, modbustcp	Serial port mode **conditions:** /defaults/ntripclient_installed = true
host	Remote host address **conditions:** mode = client
ntrip-mode **values:** auto, ntrip1, http, rtsp, udp	Transport protocol selection **conditions:** mode = ntripclient && ntripclient_installed = true
local-baudrate **values:** 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200	Serial port baud rate
local-parity **values:** none, odd, even	Parity bit
local-stop-bits **values:** 1, 2	Stop bits per serial frame
local-data-bits **values:** 8, 7	Data bits per serial frame **conditions:** mode = client mode = server mode = ntripclient
local-flowcontrol **values:** none, soft, hard	Flow control type
remote-proto **values:** raw, rfc2217	Remote side serial port protocol **conditions:** mode = client mode = server
remote-baudrate **values:** 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200	Serial port baud rate **conditions:** remote-proto = rfc2217 && mode = client
remote-parity **values:** none, odd, even	Parity bit **conditions:** remote-proto = rfc2217 && mode = client
remote-stop-bits **values:** 1, 2	Stop bits per serial frame **conditions:** remote-proto = rfc2217 && mode = client
remote-data-bits **values:** 8, 7	Data bits per serial frame **conditions:** remote-proto = rfc2217 && mode = client

remote-flowcontrol **values:** none, soft, hard	Flow control type **conditions:** remote-proto = rfc2217 && mode = client
remote-acknowledgement **values:** true, false	Require remote configuration confirmation **conditions:** remote-proto = rfc2217 && mode = client
remote-acknowledgement-timeout	Require remote configuration confirmation **conditions:** remote-proto = rfc2217 && mode = client
banner	Banner that will be sent upon connection **conditions:** remote-proto = raw && mode = client remote-proto = raw && mode = server
local-accumulation-interval **minimum:** 1	Interval (msec) between reading data from the serial port into the buffer before sending the entire buffer **conditions:** mode != modbustcp
local-accumulation-attempts	How many times to read data from serial port to accumulate buffer **conditions:** mode != modbustcp
reconnect-delay **minimum:** 1	Reconnection delay (sec)
peer-timeout **minimum:** 0	Peer inactivity disconnect timeout (sec)
bind **values:** /ip interface, /mobile modem, /tunnel eoip, /tunnel gre, /tunnel l2tp, /tunnel openvpn, /tunnel pptp, /tunnel wireguard interface, /tunnel l2tp-v3, /tunnel atunnel, /defaults server **pattern example:** bridge0	Select interface/address for incoming connections **conditions:** mode = modbustcp mode = server
port **pattern example:** 80, !80	Listening port number **conditions:** mode = modbustcp mode = server
ntrip-mountpoint	Mountpoint or other criterion for matching the data set **conditions:** mode = ntripclient && ntripclient_installed = true
ntrip-user	Dataset authentication username **conditions:** mode = ntripclient && ntripclient_installed = true
ntrip-password	Dataset authentication password **conditions:** mode = ntripclient && ntripclient_installed = true
ntrip-nmea	NMEA string for sending to server **conditions:** mode = ntripclient && ntripclient_installed = true
ntrip-bitrate **values:** true, false	Outgoing bitrate **conditions:** mode = ntripclient && ntripclient_installed = true
ntrip-initudp **values:** true, false	Sending an initial UDP packet for firewall processing **conditions:** mode = ntripclient && ntripclient_installed = true
ntrip-udpport **pattern example:** 80, !80	Local UDP port selection **conditions:** mode = ntripclient && ntripclient_installed = true

ntrip-proxyhost **pattern example:** 192.168.1.1, example.com, 192.168.1.1:80	Proxy server address:port or hostname:port (port - optional, default 2101) **conditions:** mode = ntripclient && ntripclient_installed = true
ntrip-protocol **values:** none, rts_cts, xon_xoff	Serial port flow control type **conditions:** mode = ntripclient && ntripclient_installed = true
failopen-timeout	Failopen retry timeout
debug **values:** true, false	Verbose logging for detailed diagnostics

9. service

9.1. client

The Clients section provides a centralized repository for managing user credentials for various VPN and tunneling services hosted on the router (e.g., L2TP, OpenVPN, PPTP).

Instead of defining users within each individual service's configuration, this page allows you to create a single user profile that can be associated with a specific service. This simplifies user management and ensures consistency across the platform.

Each client profile is defined by a unique username (*Name*) and a password for authentication. You can assign per-client attributes, such as a specific static *Tunnel IP* address and custom *Routes* to be pushed to the client upon connection. This enables fine-grained access control, allowing you to define precisely which network resources each remote user can access.

PROPERTIES

disabled **values:** true, false	Disable tunnel client access
service **values:** /defaults services	The services that user will be able to use
password	Client authentication password
tunnel-ip **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1/255.255.255.0	Tunnel IP address to assign to the client
route	Addresses beyond the remote side to add a local route

9.2. dhcp

9.2.1. lease

The Leases subsection provides a real-time view of all IP addresses currently assigned by the router's DHCP server. It serves as a central point for monitoring which devices are connected to your network and managing their IP address assignments.

The table displays all active leases, showing the client's MAC address, its assigned IP address, and the lease expiration time.

The primary function of this page is to manage these leases, particularly for creating **static leases** (also known as reservations). A static lease ensures that a specific device, identified by its MAC address, will

always receive the same IP address from the DHCP server. This is essential for servers, printers, or any device that requires a stable and predictable network address.

To create a reservation, you can simply select an existing dynamic lease from the list and use the **Make Static** function. This action will convert the temporary assignment into a permanent one. Additionally, for each entry, you can manage DNS integration by assigning a fully qualified domain name (FQDN).

COMMANDS

make-static	Make static lease

PROPERTIES

interface **values:** /service dhcp server, auto	Bind DHCP lease to interface for fixed IP assignments
ip **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1/255.255.255.0	Host IP address for network communication
mac **pattern example:** FE:FF:FF:FF:FF:FF	Host MAC address for hardware identification
dns **values:** true, false	Add DNS record to map hostname to IP address
fqdn **pattern example:** localhost, example.com, sample.example.com	Override DNS hostname for custom domain resolution

9.2.2. server

The Server subsection is where you define and manage individual DHCP service instances for each of your local network interfaces. This is the primary control panel for determining how IP addresses are distributed on a given network segment.

The primary configuration choice on this page is the operational **Mode**, which determines whether the router will act as a DHCP server itself or as a DHCP relay agent.

Server Mode
When set to **Server mode**, the router becomes the authoritative source for IP address assignments on a specific network. You will configure the fundamental parameters for this service, such as the *Pool Of IP Addresses* to be leased, the *Leasetime*, and essential network information to be provided to clients, including the default gateway (*Router*) and *DNS Servers*. This mode also allows for advanced configurations through custom *DHCP Options* to support specialized clients and network environments.

Relay Mode
When set to **Relay mode**, the router does not assign IP addresses itself. Instead, it listens for DHCP requests on a local interface and forwards them to a centralized, remote DHCP server. This is commonly used in enterprise networks to manage IP assignments from a single location. Configuration in this mode involves specifying the remote *Server* address and managing how DHCP options (specifically Option 82) are handled as they are relayed.

PROPERTIES

disabled **values:** true, false	Disable configuration

mode **values:** server, relay	DHCP server mode to configure request handling
server **pattern example:** 192.168.1.1, example.com, 192.168.1.1:80	DHCP relay server to forward client requests **conditions:** mode = relay
relay-mode **values:** append, replace, forward, discard	DHCP Option 82 handling for relay agent information **conditions:** mode = relay && suboption = true mode = relay && suboption = ""
remote-id	Option 82 Remote ID **conditions:** mode = relay
hops **maximum:** 255	DHCP hop limit to prevent relay loops **conditions:** mode = relay
suboption **values:** true, false	Send link selection suboption for directly connected clients **conditions:** mode = relay
debug **values:** true, false	Verbose logging for detailed diagnostics **conditions:** mode = relay
pool	IP address pool range for DHCP assignments **conditions:** mode = server
router **pattern example:** 192.168.1.1	Default gateway IP address (DHCP Option 3) **conditions:** mode = server
dns-server	DNS server IP address (DHCP Option 6) **conditions:** mode = server
ntp-server	NTP server IP address (DHCP Option 42) **conditions:** mode = server
leasetime **values:** 30m, 1h, 4h, 12h, 24h, 7d	IP lease duration to define assignment timeframe **conditions:** mode = server
flags **values:** authoritative, no-override, sequential-ip, rapid-commit	DHCP server parameter for advanced configuration **conditions:** mode = server
option **values:** tftp-server, bootfile-name, classless-static-route	Additional DHCP Options **conditions:** mode = server

9.3. dns

This section configures the router's internal DNS service, which acts as a caching DNS forwarder for the local network. It intercepts DNS queries from local devices, forwards them to upstream DNS servers, and caches the responses to accelerate subsequent lookups.

This section provides full control over the DNS resolution process:

- **DNS Forwarding Logic:** You can define the primary upstream *DNS Server*(s) for all standard queries. Additionally, the **DNS Forwarding** feature allows you to configure conditional forwarding, directing queries for specific domains (e.g., a corporate domain) to a different set of DNS servers.

- **Local Name Resolution:** The *Static-Host* feature enables you to create local DNS records, assigning easy-to-remember hostnames to devices on your network (e.g., mapping *nas.lan* to *192.168.1.10*).
- **DNS Rebind Protection:** This is a security feature that defends your local network against DNS rebinding attacks. It works by dropping upstream DNS responses that resolve to private, non-routable IP addresses (RFC1918). Exceptions for legitimate services can be configured for specific domains or for localhost-based DNS blacklists.
- **Interface Binding:** You can specify on which local *Interface*(s) the DNS server should listen for client requests.

PROPERTIES

dns-address	DNS forwarder address to resolve external queries
dns-forwarding	DNS server for this domain zone addreses resolve
interface **values:** /ip interface, /mobile modem, /tunnel eoip, /tunnel gre, /tunnel l2tp, /tunnel openvpn, /tunnel pptp, /tunnel wireguard interface, /tunnel l2tpv3, /tunnel atunnel, /defaults server	DNS interface to process name resolution requests
static-host	Static DNS A-record for hostname-to-IP mapping
rebind-protection **values:** true, false	DNS rebind protection to block private IP responses
rebind-localhost **values:** true, false	Allow localhost DNS responses for testing purposes **conditions:** rebind-protection = true
rebind-domain	Allow RFC1918 private IP responses for internal domains **conditions:** rebind-protection = true

9.4. l2tp server

This section configures the router's built-in L2TPv2 server, allowing remote users to establish secure tunnels into the local network. The fundamental setup involves defining the server's own *IP Address* within the tunnel and an *IP Pool* from which addresses will be assigned to connecting clients.

The most critical configuration choice is the **Encryption** method, which determines the security level of the tunnel:

- **IPsec (Recommended):** For maximum security, you should use L2TP over IPsec (*l2tp/ipsec*). When you select this option, the router streamlines the configuration by automatically generating the required IPsec connection and firewall rules.

After selecting *ipsec* encryption and applying the settings, you must navigate to the *Service / IPsec* section to set a **Pre-Shared Key** or configure another IPsec authentication method for the auto-generated connection. The tunnel will not be fully functional until this step is completed.

- **MPPE (Microsoft Point-to-Point Encryption):** This provides a simpler, integrated encryption method that is automatically paired with MS-CHAPv2 authentication. It is suitable for clients that do not support IPsec.
- **None:** An unencrypted L2TP tunnel. This option is strongly discouraged for use over public networks.

Authentication for connecting users is managed centrally. The L2TP server will authenticate clients against the user profiles defined in the *Service / Clients* section.

PROPERTIES

disabled **values:** true, false	Disable configuration
ip-addr **pattern example:** 192.168.1.1	Local tunnel IP address
ip-pool	Client IP address assignment range
auth **values:** any, chap, mschap, mschap-v2, pap, eap	Authentication protocol **conditions:** encryption = mppe
encryption **values:** none, mppe	Tunnel encryption method **conditions:** /defaults/ipsec_installed = true
psk **minLength:** 8	IPsec pre-shared key for tunnel authentication **conditions:** encryption = ipsec
ppp-option **values:** lcp-echo-failure, lcp-echo-interval, lcp-max-configure, lcp-max-failure, lcp-max-terminate, lcp-restart, ipcp-accept-local, ipcp-accept-remote, ipcp-max-configure, ipcp-max-failure, ipcp-max-terminate, ipcp-restart, mru, mtu	Point-to-Point Protocol Daemon (pppd) options
debug **values:** true, false	Verbose logging for detailed diagnostics

9.5. drm client

This section configures the client that connects the router to the **Donyx DRM**, a centralized platform for remote fleet management. By establishing a secure, persistent connection to the central server, this agent enables remote monitoring, configuration management, and troubleshooting of the device from a single dashboard.

The configuration involves specifying the *Server* address that the router will connect to. The connection's health is maintained via periodic *Keepalive* messages, and the agent regularly pushes its full status to the server based on the *Force Update* interval.

All communication is secured using the provided *Encryption Key* to ensure data integrity and confidentiality.

PROPERTIES

disabled **values:** true, false	Disable configuration
server **pattern example:** 192.168.1.1, example.com, 192.168.1.1:80	Address of management server DRM
update-interval	Full status report interval (sec)
keep-alive-interval	Keepalive heartbeat interval (sec)
key	SHA-256 data encryption key

9.6. ntp

This section configures the Network Time Protocol (NTP) service, which is responsible for maintaining accurate system time on the router. Accurate time is a prerequisite for reliable logging, valid certificate-based authentication, and the proper operation of many network protocols.

The router can function in two distinct NTP roles:

- **NTP Client (Time Synchronization)** -
  In its primary role as an NTP client, the router synchronizes its own clock with external, authoritative time servers. You specify these upstream servers in the *Pool* list. The router will periodically query these servers to ensure its internal clock remains accurate.
- **NTP Server (Time Provider for LAN)** -
  By enabling the *Server* option, the router itself becomes an NTP server for your local network. Client devices on your LAN can then be configured to use the router's IP address as their time source. This ensures all devices on your local network are perfectly synchronized, and it reduces the amount of NTP traffic going out to the internet.

Monitoring Synchronization Status
The **Info** function provides access to a detailed status table of the router's connections to its upstream NTP servers.

COMMANDS

info	NTP client status information

PROPERTIES

disabled **values:** true, false	Disable configuration
pool	Network Time Protocol (NTP) server pool for clock synchronization
server **values:** true, false	Enable Network Time Protocol (NTP) server mode

9.7. openvpn server

This section allows you to configure the router as a robust and highly configurable OpenVPN server. OpenVPN creates a secure, encrypted tunnel over the internet, enabling secure remote access for clients or establishing site-to-site connections.

The configuration is comprehensive, allowing you to control every aspect of the server's operation.

Public Key Infrastructure (PKI) and Keys

Secure operation relies on a set of certificates and keys:

- **CA Certificate:** The *Ca* file is the root of trust for your VPN. All client and server certificates must be signed by this Certificate Authority.
- **Server Certificate:** The *Cert* is the server's public certificate, presented to clients to prove its identity.

- **TLS Authentication Key:** The *Tls-Auth* key provides an additional layer of HMAC authentication on the control channel, hardening the server against DoS attacks and port scanning. You can generate a new key using the **Generate TA Key** function. This key is stored in *Storage / File* and must be shared with all clients.

User Authentication

OpenVPN supports two primary methods for authenticating clients:

1. **Certificate-Only Authentication:** Access is granted solely based on a valid client certificate signed by the server's CA. No username or password is required.
2. **Certificate and Username/Password Authentication:** This provides two-factor authentication. Access is granted only if **both** a valid client certificate is presented **and** the provided username and password are correct. The validation logic is as follows:
    - If the username does not exist on the router, access is **denied**.
    - If the username exists but the password does not match, access is **denied**.
    - If both username and password are correct, access is **granted**.

To enable username/password authentication for a user, you **must** create a corresponding profile for them in the *Service / Clients* section, ensuring you set a password for that profile.

COMMANDS

generate-ta-key	Generate a key file used as TA Key or Static Key filename - Configuration name (required)

PROPERTIES

disabled **values:** true, false	Disable configuration
dev-type **values:** tun, tap	Virtual interface type
protocol **values:** tcp, udp	Tunnel transport protocol
port **pattern example:** 80, !80	Listening port number
tunnel-ip **pattern example:** 192.168.1.1/24	Local tunnel IP address
pool **pattern example:** 192.168.1.100, 192.168.1.1-192.168.1.100	Client IP address assignment range
cipher **values:** AES-128-CBC, AES-128-GCM, AES-192-CBC, AES-192-GCM, AES-256-CBC, AES-256-GCM	Data channel encryption algorithm
auth **values:** MD5, SHA1, SHA256, SHA384, SHA512	Authentication algorithm
ta-key **values:** /storage file	Hash-based Message Authentication Code (HMAC) key for packet validation

ca **values:** /storage certificate	Certificate Authority (CA) public key
cert **values:** /storage certificate	Local Transport Layer Security (TLS) certificate bundle (cert + key)
keepalive **pattern example:** 10 60	Configure keepalive ping and auto-restart on link failure
push **values:** route, route-gateway, route-metric, route-delay, redirect-gateway, ip-win32, dhcp-option, inactive, ping, ping-exit, ping-restart, setenv, auth-token, persist-key, persist-tun, topology, echo, comp-lzo, socket-flags, sndbuf, rcvbuf	Push configuration parameters to client
flag **values:** allow-recursive-routing, auth-nocache, client-to-client, comp-noadapt, disable, duplicate-cn, fast-io, float, mtu-test, multihome, ncp-disable, nobind, opt-verify, passtos, persist-key, persist-local-ip, persist-remote-ip, persist-tun, ping-timer-rem, pull, push-peer-info, push-reset, remote-random, route-nopull, single-session, suppress-timestamps, tcp-nodelay, tls-client, tls-exit, tls-server, username-as-common-name	Tunnel advanced configuration flags
extra **values:** auth-retry, bcast-buffers, comp-lzo, compress, connect-freq, connect-retry, connect-retry-max, connect-timeout, ecdh-curve, explicit-exit-notify, fragment, hand-window, hash-size, ifconfig-pool-netmask, ifconfig-push-local, ifconfig-push-netmask, inactive, key-direction, keysize, link-mtu, max-clients, mssfix, mtu-disc, ping, ping-exit, ping-restart, prng, pull-filter-accept, pull-filter-ignore, pull-filter-reject, rcvbuf, reneg-bytes, reneg-pkts, reneg-sec, replay-persist, replay-window, resolv-retry, shaper, sndbuf, tcp-queue-limit, tls-timeout, tls-version-min, tran-window, tun-mtu, tun-mtu-extra, txqueuelen, verb, verify-client-cert, verify-x509-name, x509-username-field	Tunnel extra parameters (custom options)

9.8. pinger

The Pinger section configures an automated service for two primary purposes: general-purpose connection monitoring and dynamic management of WAN failover.

As a monitoring tool, it can be configured to periodically test connectivity to any target host using various probe types (ICMP, TCP, etc.). This allows you to check the status of any critical link or service.

Its primary automated action, however, is to manage routing priority for failover.

When the Pinger detects that the quality of such a route has degraded—based on user-defined thresholds — it automatically **increases the route's metric**.

It is crucial to understand that this automated metric adjustment is specifically and exclusively applied to **interfaces that are configured as a *Default Route***.

This action de-prioritizes the underperforming link, causing the router's routing engine to automatically prefer an alternate default route with a better (lower) metric.

PROPERTIES

disabled **values:** true, false	Disable configuration
type **values:** icmp, arping, http, tcp	Reachability check method

host	Endpoint for availability check **conditions:** type = tcp
interval **minimum:** 10	Interval between checks
retries **minimum:** 1	Limit of failed checks before restarting the interface
count **minimum:** 1	Number of packets to check per attempt **conditions:** type = arping type = http type = icmp
size **minimum:** 1, **maximum:** 65535	Packet size **conditions:** type != http
rtt-threshold	Round-trip time (RTT) failure threshold
loss-threshold	Packet loss failure threshold **conditions:** type = icmp type = arping

9.9. pptp server

This section configures the router's built-in PPTP (Point-to-Point Tunneling Protocol) server. It allows remote users to establish a VPN tunnel into the local network.

The fundamental setup involves defining the server's own *IP Address* within the tunnel and an *IP Pool* from which addresses will be assigned to connecting clients.

The most critical configuration choice is the **Encryption** method, which determines the security level of the tunnel:

- **IPsec (Recommended for Security):** This option encapsulates PPTP traffic within a secure IPsec tunnel. To simplify this complex setup, the router will **automatically generate** the necessary configuration when this mode is enabled:
  - An IPsec Connection (*pptp_server_connection*) and Association (*pptp_server_association*) in the *Service* / *IPsec* section.
  - An accompanying firewall rule in *Firewall* / *Filter* (*pptp_server_autogenerated*).

For the IPsec tunnel to function correctly, a manual step is required. After applying the settings, you **must** navigate to *Service* / *IPsec* / *Connection*, select the *pptp_server_connection* entry, and configure an authentication method (e.g., set a Pre-Shared Key). The connection will not work until this is done.

- **MPPE (Microsoft Point-to-Point Encryption):** This provides a simpler, integrated encryption method that automatically pairs with the MS-CHAPv2 authentication protocol.
- **None:** An unencrypted PPTP tunnel. This option is **strongly discouraged** for use over public networks as it provides no data confidentiality.

User authentication is managed centrally in the *Service* / *Clients* section.

When specifying a range for the **IP Addresses** and **IP Pool** fields, be aware of the specific syntax conversion. For example:

• 1.1.1.2-1.1.1.10 → 1.1.1.2-10

• 1.1.2.3-1.1.5.10 → 1.1.2-5.10 (representing the range from 1.1.2.10 to 1.1.5.10)

PROPERTIES

disabled **values:** true, false	Disable configuration
ip-addr	Local tunnel IP address
ip-pool	Client IP address assignment range
auth **values:** any, chap, mschap, mschap-v2, pap, eap	Authentication protocol **conditions:** encryption = mppe
encryption **values:** none, mppe	Tunnel encryption method **conditions:** /defaults/ipsec_installed = true
psk **minLength:** 8	IPsec pre-shared key for tunnel authentication **conditions:** encryption = ipsec
ppp-option **values:** lcp-echo-failure, lcp-echo-interval, lcp-max-configure, lcp-max-failure, lcp-max-terminate, lcp-restart, ipcp-accept-local, ipcp-accept-remote, ipcp-max-configure, ipcp-max-failure, ipcp-max-terminate, ipcp-restart, mru, mtu	Point-to-Point Protocol Daemon (pppd) options
debug **values:** true, false	Verbose logging for detailed diagnostics

9.10. snmp

This section configures the router's built-in SNMP (Simple Network Management Protocol) agent, which allows the device to be monitored by a central Network Management System (NMS). SNMP provides standardized access to a wide range of operational data, such as CPU load, memory usage, interface statistics, and cellular signal strength.

The router supports two major versions of the protocol, each offering a different level of security.

SNMPv2c
This is a simpler, widely used version that relies on a *Community* string for access control. This string acts as a shared password between the router and the NMS.

The community string is transmitted in clear text. Therefore, SNMPv2c should only be used in trusted, secure networks.

SNMPv3
This version provides a robust and modern security framework based on user authentication and data encryption (USM - User-based Security Model). The security level is highly configurable.

Writable Access

The *Writable* option, available for both versions, allows the NMS not only to read data (GET requests) but also to send commands to modify the router's state (SET requests), such as changing GPIO states. This option should be enabled with extreme caution and only when necessary.

PROPERTIES

disabled **values:** true, false	Disable configuration
version **values:** v2c, v3	SNMP protocol version
port **pattern example:** 80, !80	Listening port number for incoming SNMP requests
community **pattern example:** bridge0, vpn-client1, user_1@example.com	SNMP community string for authentication
sys-name **pattern example:** bridge0, vpn, client_1	Device name for identification in the monitoring system
sys-contact	Device contact details for identification in the monitoring system
sys-location	Device location for identification in the monitoring system
sys-description	Description of the device for identification in the monitoring system
username	Client authentication username **conditions:** version = v3
auth-passphrase **minLength:** 8	Passphrase (SHA) for authentication **conditions:** version = v3
priv-passphrase **minLength:** 8	Passphrase (SHA) for traffic encryption **conditions:** version = v3
security-level **values:** noauth, auth, priv	The level of security used for communication **conditions:** version = v3
writable **values:** true, false	Enable read-write access

9.11. vrrp

This section configures the Virtual Router Redundancy Protocol (VRRP), a standard protocol designed to provide high availability for the default gateway. It achieves this by grouping two or more routers into a single "virtual router," which presents a consistent, virtual IP address to all clients on the network.

Within this group, one router is elected as the **Master**, actively handling traffic, while the others act as **Backups**. This election is determined by the *Priority* value (a higher value is more preferred). If the Master router fails, the highest-priority Backup router automatically and seamlessly takes over its role.

PROPERTIES

disabled **values:** true, false	Disable configuration
virtual-ip	Virtual default gateway IP address
virtual-mac **values:** true, false	Assign virtual MAC address to interface
preemption-mode **values:** true, false	Enable VRRP master preemption

virtual-id **minimum:** 1, **maximum:** 255	VRRP virtual router ID
priority **minimum:** 1, **maximum:** 255	VRRP router priority
delay **minimum:** 1, **maximum:** 3600	Interval between sending VRRP-advertise messages (sec)
auth **values:** none, pw, ah	VRRP authentication type
key **minLength:** 1, **maxLength:** 8	Passphrase for VRRP packet authentication **conditions:** auth = ah auth = pw

10. storage

10.1. certificate

This section provides a toolkit for end-to-end management of TLS credentials (certificates and private keys). These credentials are essential for securing services hosted on the router, such as the web interface (HTTPS), OpenVPN, IPsec, and more.

Generating and Acquiring Credentials

You can obtain certificates for the router in several ways:

- **Create CSR (Certificate Signing Request):** This is the standard method for obtaining a certificate from a trusted public Certificate Authority (CA). You generate a CSR, send it to the CA, and they will return a signed certificate, which you can then import.
- **Create X.509:** This option generates a **self-signed certificate**. It is useful for internal networks, testing, or scenarios where a publicly trusted certificate is not required.
- **Sign CSR:** This allows the router itself to act as a miniature Certificate Authority. You can take a CSR from another device (e.g., another router or a client) and sign it using a CA certificate stored on this router.
- **Import Certificate:** Use this to upload existing certificates and private keys that were generated externally.

Managing and Deploying Credentials

Once a certificate is in the store, you can manage it:

- **Export Certificate:** This allows you to back up your credentials or distribute them to clients. The export is done in the standard, password-protected PKCS12 format, which can bundle the certificate, its private key, and any intermediate certificates into a single file.

COMMANDS

cert-create	Generate certificate for secure service authentication name - New filename for certificate storage (required) ca - Certificate Authority (CA) certificate authority - The certificate can be used to sign pkey-size - Private key size to configure cryptographic strength cn - Certificate common name (required) alt-name - Certificate subject alternative name org - Subject organization unit - subject organizational unit state - Subject state country - Subject country location - Subject location days - Certificate validity period for expiration management
csr-create	Generate Certificate Signing Request (CSR) file for third-party validation name - CSR filename for signing request identification (required) pkey-size - Public key size to match cryptographic key pair passphrase - Certificate file passphrase to encrypt private key storage cn - Certificate common name (required) alt-name - Certificate subject alternative name org - Subject organization unit - subject organizational unit state - Subject state country - Subject country location - Subject location
csr-sign	Sign Certificate Signing Request to validate domain ownership name - Certificate Signing Request file for authority validation (required) ca - Certificate Authority (CA) certificate (required) alt-name - Certificate subject alternative name days - Signed certificate validity period to enforce expiration dates passphrase - Certificate file passphrase to encrypt private key storage
cert-export	Export PKCS#12 certificate bundle for backup or migration name - Certificate file for export (required) include-pkey - Export with private-key passphrase - File passphrase if there is such additional-cert - Include additional certificate
cert-import	Import certificate file with optional name and decryption passphrase name - Certificate file to import for TLS configuration (required) passphrase - Certificate file passphrase to encrypt private key storage
rename	Rename certificate file for organizational purposes new-name - New filename for certificate storage (required)

PROPERTIES

issuer	The issuer name
subject	The subject name
alt-name	The subject alternative name **conditions:** alt-name != ""
not-before	The date before which certificate is invalid
not-after	The date after which certificate wil be invalid
fingerprint	The certificate hash

10.2. file

This section provides a file management interface, allowing for the direct manipulation of files stored on the router. It serves as the primary tool for uploading, downloading, and organizing files required for various system operations, such as configuration backups, firmware images, or custom scripts.

The file manager distinguishes between two primary storage locations:

- **Temporary Storage:** Files stored here reside in volatile memory (RAM) and will be lost upon a system reboot. This location is typically used as a staging area for files before they are installed or imported by a specific service.
- **Persistent Storage:** Files stored here are written to the router's non-volatile flash memory and will persist across reboots.

From this interface, you can upload new files, download existing ones, move them between storage locations, or delete them. Additionally, you can manage file permissions by toggling the *Executable* flag for scripts.

COMMANDS

upload	Upload files to the temporary storage
download	Download file from the device storage
import	Import file to the device storage

PROPERTIES

file-type	File type
file-size	File size
last-change	File last modification timestamp
last-access	File last access timestamp

11. system

11.1. access

11.1.1. radius

This section enables centralized authentication and authorization via the **Remote Authentication Dial-In User Service (RADIUS)** protocol.

In this architecture, the router acts as a **Network Access Server (NAS)**. When a user attempts to access the device, the router transmits the credentials to a dedicated RADIUS server for validation. Successful communication requires specific identification of the NAS and a valid secret for server-side authentication.

PROPERTIES

disabled **values:** true, false	Disable Radius authentication
server **pattern example:** 192.168.1.1, example.com, 192.168.1.1:80	Authentication server
nas-id **maxLength:** 32 **pattern example:** bridge0, vpn-client1, user_1@example.com	Unique NAS Identifier
secret	Password for NAS authentication on the RADIUS server

11.1.2. ssh

This section configures the router's built-in SSH server, which provides secure, encrypted command-line access for advanced administration, configuration, and diagnostics.

In addition to enabling or disabling the service and changing the listening *Port*, this page offers granular control over the cryptographic algorithms used by the SSH server. This allows administrators to harden the device's security posture by disabling weaker algorithms and ensuring compliance with specific corporate or regulatory security policies.

The following cryptographic components can be customized:

- **Cipher:** Defines the symmetric encryption algorithms for data confidentiality.
- **Host Key Algorithms:** Determines the public key algorithms the server can use to prove its identity.
- **KEX Algorithms:** Controls the methods used to securely generate and exchange session keys.
- **MAC Algorithms:** Specifies the hashing algorithms used to ensure message integrity and authenticity.

Modifying these cryptographic settings is intended for advanced users. An incorrect configuration may result in the inability to connect to the router via SSH. Only adjust these parameters if required to meet a specific security policy.

PROPERTIES

disabled **values:** true, false	Disable configuration
port **pattern example:** 22, 51820	Secure Shell (SSH) service listening port
cipher **values:** chacha20-poly1305@openssh.com, aes128-ctr, aes192-ctr, aes256-ctr, aes128-gcm@openssh.com, aes256-gcm@openssh.com	Secure Shell (SSH) encryption cipher selection
host-key-alg **values:** rsa-sha2-512, rsa-sha2-256, ssh-rsa, ssh-ed25519	Secure Shell (SSH) Message Authentication Code (MAC) algorithm selection
kex-alg **values:** curve25519-sha256, curve25519-sha256@libssh.org, ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521, diffie-hellman-group-exchange-sha256, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, diffie-hellman-group14-sha256	Secure Shell (SSH) key exchange method selection
mac-alg **values:** umac-64-etm@openssh.com, umac-128-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-512-etm@openssh.com, hmac-sha1-etm@openssh.com, umac-64@openssh.com, umac-128@openssh.com, hmac-sha2-256, hmac-sha2-512, hmac-sha1	Secure Shell (SSH) host key algorithm selection

11.1.3. web

This section configures access to the router's web-based management interface. It allows you to control the protocols (HTTP/HTTPS) and ports used for administrative access.

The primary configuration is enabling secure access via **HTTPS** by the *Use HTTPS* function. This encrypts all communication between your browser and the router, protecting your login credentials and configuration data from interception. While standard **HTTP** access is available, it is unencrypted and should only be used on completely trusted networks.

To enable HTTPS, you must select the *CERT File* (the server's public certificate) and the *KEY File* (its corresponding private key) from the certificate store. Using a custom, trusted certificate eliminates browser security warnings and ensures a secure administrative session.

For additional security hardening, you can:

- Change the listening *Port* from the default value to a non-standard one.
- Specify the *TLS Minimum* version to enforce the use of modern, secure cryptographic standards.

PROPERTIES

use-http **values:** true, false	Enable HTTP access to allow unencrypted web interface connections
http-port **pattern example:** 80, !80	HTTP listening port for incoming connection requests **conditions:** use-http = true
use-https **values:** true, false	Enable HTTPS access to secure web interface connections
https-port **pattern example:** 80, !80	HTTPS listening port for encrypted connection requests **conditions:** use-https = true
cert **values:** /storage certificate	Certificate for encryption and identity validation **conditions:** use-https = true
tls **values:** 1.1, 1.2, 1.3	Minimum allowed TLS version for secure protocol negotiation **conditions:** use-https = true
redirect-to-https **values:** true, false	Redirect HTTP traffic to HTTPS port for enforced encryption **conditions:** use-https = true && use-http = true

11.2. block dev

This section provides low-level management for block storage devices and serves as the primary interface for configuring the **overlay filesystem**.

The *Overlay_dev* function allows for selecting a storage device to act as the primary overlay. This effectively expands the system's root partition, enabling the installation of additional software packages and the storage of larger configuration files beyond the capacity of the internal flash memory.

Using the overlay feature with an external drive is an advanced operation with significant consequences. Please review the following points carefully before proceeding:

- **Formatting and Data Loss:** When an external drive is selected to be used as the overlay, it **will be formatted**, erasing all existing data on it.
- **Package Loss on Reversal:** If the system is switched from an external overlay back to the internal storage, all packages that were installed on the external drive will be lost.

- **System Reboot on Removal:** Physically removing an external drive while it is in use as the primary overlay will trigger an immediate system reboot.

The following file systems are supported for SD cards: **FAT32, EXT2, EXT3, EXT4, F2FS, JFFS2**, and **NTFS**.

The **exFAT** and **BFS** file systems are not supported. Before using an SD card in the router, ensure that it is formatted with one of the supported file systems.

The system will utilize **only the first partition** of the selected storage device. It is essential that this partition's capacity is sufficient for the intended purpose (e.g., for installing packages or storing data).

COMMANDS

overlay_dev	Choose device for mount as overlay device - The device name (required)

11.3. config

This section provides tools for managing the router's configuration. The system utilizes a two-stage configuration model: changes are first made to a temporary "candidate" configuration and must be explicitly committed before they become part of the permanent "running" configuration.

The configuration encompasses the core set of parameters for the router's operation, including sensitive data such as passwords, keys, hashes, and certificates required by various services.

Importing a configuration from an incompatible firmware version or a different device model can lead to system instability or render the device inaccessible after the changes are committed.

COMMANDS

revert	Revert all uncommitted configuration changes
commit	Commit the current running configuration
export	Export the current running configuration
import	Import configuration from file file - Configuration file to import (required)
default	Reset to default settings

11.4. logging

This section configures the behavior of the system logging service, which records all significant operational events, from user logins and configuration changes to errors and system alerts. This page allows you to control the destination and storage of these critical log messages.

The logging can be directed to two primary destinations:

- **Remote Syslog Server:** This feature enables real-time streaming of log messages to a remote server. You can specify the server's *Address* and the *Transport* protocol (e.g., UDP or TCP). The *Prefix* option allows for easy identification of messages from this specific device on the central server.
- **Local File:** Alternatively, you can enable *Log To File* to store logs locally on the router. This is useful for on-device diagnostics and troubleshooting, especially when a remote server is not available.

The log file size displayed in the */storage/file* section may appear as zero, even if logging to a file is active. This is because the value is not updated in real-time. To force a refresh and view the current size, it is necessary to click on the ***file*** part of the navigation path directly. This action triggers a schema update and repopulates the interface with the current data.

PROPERTIES

remote-host **pattern example:** 192.168.1.1, example.com, 192.168.1.1:80	Remote host for receiving log
remote-proto **values:** tcp, udp	Transport protocol for sending log
remote-prefix	Prefix for log entries
file-log **values:** true, false	Enable logging to local file
file-name	Log file name **conditions:** file-log = true
file-size **minimum:** 1, **maximum:** 8192	Maximum file size (Kbyte) **conditions:** file-log = true

11.5. management

This section is the central hub for configuring fundamental system parameters and performing essential maintenance operations.

It allows you to define the router's core identity (such as its hostname and time zone) and execute system-wide commands. From here, you can reboot the device, upgrade its firmware, perform a factory reset, or generate a diagnostic report for troubleshooting.

COMMANDS

change	Set system date and time manually isotime - Set local time isodate - Set local date
report	Generate system diagnostics report
reboot	Reboot device
upgrade	Upgrade firmware file - File in temporary storage (required) reset - Reset to default settings

PROPERTIES

hostname **pattern example:** localhost, example.com, sample.example.com	Device hostname
timezone **values:** UTC-12, UTC-11, UTC-10, UTC-9, UTC-8, UTC-7, UTC-6, UTC-5, UTC-4, UTC-3, UTC-2, UTC-1, UTC, UTC+1, UTC+2, UTC+3, UTC+4, UTC+5, UTC+6, UTC+7, UTC+8, UTC+9, UTC+10, UTC+11, UTC+12, UTC+13, UTC+14	Device time zone

11.6. package

This section provides an interface for managing the software packages installed on the router. It allows you to extend the router's functionality by installing new applications or libraries.

The main view displays a list of all currently installed packages, along with key details such as their *version*, *dependencies*, *size*, and a short *description*.

Package Management

- **Installing a Package:** The *Install* allows you to select a package file (e.g., in *.ipk* format) that has been previously uploaded to the router's temporary storage.
- **Removing a Package:** The *Clean* is used to uninstall a selected package from the system.

COMMANDS

install	Install software package file - File in temporary storage (required)

PROPERTIES

version	Version of the installed package
depends	Package dependencies
size	Disk space usage by the package
description	Package description from maintainer

11.7. user

This section is used to create, manage, and define access rights for all user accounts that can log into the router's management interface.

The system provides comprehensive control over both **authentication** (how users prove their identity) and **authorization** (what they are permitted to do after logging in).

Authentication Methods

Two primary methods of authentication are supported:

- **Password:** Traditional password-based login. A user's password can be set or changed here.
- **SSH Key:** More secure, passwordless authentication using public key cryptography. A user's public SSH key can be imported, allowing them to log in via SSH without a password.

Authorization and Access Control

Access rights are managed through a role-based model, which can be further refined with specific restrictions:

- **Access Level:** Each user is assigned a high-level role:
- *administrator*: Has full, unrestricted access to all system settings.
- *user*: Has limited, read-only access by default, which can be customized.

- **Denied Menus:** For more fine-grained control, user access to specific menu items can be explicitly blocked by adding them to the *denied* list.

Account Management

In addition to defining access rights, a user's access can be temporarily revoked without deleting their profile.

COMMANDS

password sha256	Change user password new - Password (plain text or hash [md5 sha512], like in /etc/shadow)
import-ssh-key	Import a user's SSH key file - SSH key file in temporary storage

PROPERTIES

disabled **values:** true, false	Disable user account and access
denied **values:** /defaults schemas	Menu items not accessible to the user
level **values:** administrator, user	User access level
ssh-keys **values:** $ssh-keys	User SSH key **conditions:** ssh-keys != ""

12. tools

This section provides a suite of on-demand utilities for network diagnostics and system troubleshooting. Unlike configuration sections, the tools here perform one-off actions to analyze and resolve issues in real-time.

The available utilities can be used to perform a range of diagnostic tasks, such as testing network reachability at different layers, performing traffic analysis, initiating client sessions to remote hosts, and directly accessing system logs or downloading files.

COMMANDS

download	Upload file to device storage url - Resource URL (required) to-file - Save file with specified filename username - Username for authentication password - Password for authentication
ping	ICMP echo requests to remote host host - Host address (required) interface - Network interface count - Number of packets to send size - Packet size

sniffer	Network packet analyzer interface - Packet capture interface (required) proto - Protocol filter src-addr - Source IP address filter dst-addr - Destination IP address filter src-port - Source port filter dst-port - Destination port filter src-mac - Source MAC address filter dst-mac - Destination MAC address filter append - Logical operator for filters count - Number of packets to send file - File name for packet dump
arping	ARP requests to remote host host - Host address (required) interface - Network interface (required) count - Number of packets to send
traceroute	Traceroute to remote host host - Host address (required) max_hops - Maximum number of hops to host protocol - Use ICMP echo or UDP datagrams port - Application port
telnet	Telnet client session to remote host host - Host address (required) interface - Network interface port - Application port
journal	View system log entries filter - Regular expression for filtering output last - Show last N entries
dashboard	System summary
ssh	SSH client session to remote host host - Host address (required) username - Client authentication username (required) port - Application port

13. tunnel

13.1. eoip

This section configures Ethernet over IP (EoIP) tunnels, a protocol for creating a transparent Layer 2 network bridge between two remote locations over an IP network.

An EoIP tunnel works by encapsulating full Ethernet frames into IP packets, effectively creating a virtual point-to-point Ethernet link between two routers. The primary use case is to bridge two separate LAN segments seamlessly into a single broadcast domain, making them behave as if they were connected by a physical Ethernet cable.

Originally developed by MikroTik, EoIP is a straightforward and efficient solution for L2 network extension. These tunnels can be established over any underlying IP transport, including other VPNs like IPsec or OpenVPN, providing a flexible way to extend Layer 2 connectivity securely.

PROPERTIES

disabled **values:** true, false	Disable configuration
local-ip **values:** /ip interface, /mobile modem, /tunnel eoip, /tunnel gre, /tunnel l2tp, /tunnel openvpn, /tunnel pptp, /tunnel wireguard interface, /tunnel l2tpv3, /tunnel atunnel, /defaults server, auto **pattern example:** bridge0, 192.168.1.1	Local endpoint address or source interface

remote-ip **pattern example:** localhost, sample.example.com, 8.8.8.8	Remote endpoint (FQDN or IP address)
tunnel-ip **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1/255.255.255.0	Tunnel interface IP address for virtual networking
tunnel-id **minimum:** 1, **maximum:** 65535	Tunnel identifier
macaddr **pattern example:** FE:FF:FF:FF:FF:FF	MAC address for hardware identification
mtu **minimum:** 70, **maximum:** 65535	Tunnel interface MTU to define maximum packet size
ttl **minimum:** 1, **maximum:** 255	Tunnel interface TTL to control packet lifetime
dscp	Differentiated Services Code Point (DSCP) value to prioritize network traffic
encryption **values:** none	Tunnel encryption method for secure data transmission
psk **minLength:** 8	IPsec pre-shared key for tunnel authentication **conditions:** encryption = ipsec

13.2. gre

This section configures Generic Routing Encapsulation (GRE) tunnels, a versatile protocol for carrying a wide range of network layer protocols over point-to-point IP tunnels.

GRE's primary strength is its ability to transport traffic that is not natively routable over a standard IP network, such as non-IP protocols or multicast traffic. It is also commonly used to create overlay networks, for example, to transport IPv6 traffic over an IPv4-only network (and vice versa).

This section lets you configure all standard GRE parameters, including tunnel endpoints (*Local IP* and *Remote IP*) and an optional *Key* used to identify a specific tunnel when multiple GRE connections exist to the same remote peer. It also includes advanced controls for packet handling, such as *TTL* and *ToS* manipulation, and a built-in *Keepalive* mechanism to monitor the tunnel's health and automatically tear it down if the remote endpoint becomes unreachable.

PROPERTIES

disabled **values:** true, false	Disable configuration
local-ip **values:** /ip interface, /mobile modem, /tunnel eoip, /tunnel gre, /tunnel l2tp, /tunnel openvpn, /tunnel pptp, /tunnel wireguard interface, /tunnel l2tpv3, /tunnel atunnel, /defaults server, auto **pattern example:** bridge0, 192.168.1.1	Local endpoint address or source interface
remote-ip **pattern example:** localhost, sample.example.com, 8.8.8.8	Remote endpoint (FQDN or IP address)
tunnel-ip **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1/255.255.255.0	Tunnel interface IP address for virtual networking
key **minimum:** 0, **maximum:** 4294967295	GRE tunnel key for session identification

encryption **values:** none	Tunnel encryption method for secure data transmission
psk **minLength:** 8	IPsec pre-shared key for tunnel authentication **conditions:** encryption = ipsec
mtu **minimum:** 70, **maximum:** 65535	Tunnel interface MTU to define maximum packet size
ttl **minimum:** 1, **maximum:** 255	Tunnel interface TTL to control packet lifetime
dscp	Differentiated Services Code Point (DSCP) value to prioritize network traffic
do-not-frag **values:** true, false	Set IP Don’t Fragment (DF) flag
keepalive-delay **minimum:** 0, **maximum:** 4294967	Tunnel keepalive interval for connection persistence
keepalive-retries **minimum:** 0, **maximum:** 4294967295	Keepalive retry count before failure **conditions:** keepalive-delay != "" && keepalive-delay != 0

13.3. ipsec

13.3.1. association

This section is where the core IPsec security policies are defined. These policies, often referred to as Child SAs in IKEv2 terminology, are the rules that determine precisely which network traffic gets protected, how it is protected, and between which peers.

The primary function here is to configure the *Local* and *Remote traffic selectors*. These selectors define the "interesting traffic" that will trigger the IPsec encapsulation. By specifying IP ranges, protocols, and ports, you can create highly specific rules that match your network’s security requirements.

Each policy is bound to a specific *Connection* (which defines the peers) and uses a *Proposal* (which defines the Phase 2 encryption and integrity algorithms), creating a modular and scalable configuration.

The *Apply* function in this section performs a soft reload, allowing for the addition or modification of policies without disrupting existing, active Security Associations. To manually terminate a specific SA, use the status page.

PROPERTIES

disabled **values:** true, false	Disable configuration
connection	IKE Phase 1 configuration for SA negotiation
proposal **values:** /tunnel ipsec proposal	ESP encryption and authentication parameters
auto **values:** route, start	Action to perform after applying the configuration
type **values:** tunnel, transport, passthrough, drop	IPSec Mode Negotiation for SA
leftsubnet	Traffic source for agreeing of traffic selectors permitted for the tunnel

rightsubnet	Traffic destination for agreeing of traffic selectors permitted for the tunnel

13.3.2. connection

This section is where the IKE (Internet Key Exchange) connections are defined. An IKE connection establishes the identity and security parameters for the peers themselves, creating the secure control channel (the Phase 1 SA) that is used to negotiate the actual data encryption policies (Phase 2 Child SAs).

The core configuration involves:

- Defining the peers by specifying their *local* and *remote* gateway addresses.
- Selecting the IKE version (*IKEv1* or *IKEv2*) that will govern the negotiation process.
- Specifying the peer identities (*Local ID*, *Remote ID*) and the authentication method (*Local auth*, *Remote auth*). This is the heart of the security model, supporting methods like Pre-Shared Keys (PSK) and public key (certificate-based) authentication.
- Assigning an IKE Profile that contains the specific Phase 1 encryption and integrity algorithms to be used for protecting the control channel.

The *Apply* function in this section performs a soft reload, allowing for the addition or modification of policies without disrupting existing, active Security Associations. To manually terminate a specific SA, use the status page.

PROPERTIES

left **values:** /ip interface, /mobile modem, auto	Local endpoint address or source interface
right **pattern example:** localhost, sample.example.com, 8.8.8.8	Remote endpoint (FQDN or IP address)
keyexchange **values:** ikev2, ikev1	IKE protocol version selection
profile **values:** /tunnel ipsec profile	IKE/ISAKMP encryption and authentication proposal
leftid	Local IKE identity for authentication **conditions:** auth = pubkey
rightid	Remote IKE identity for verification
auth **values:** psk, pubkey	IKE authentication method
psk **minLength:** 8	IKE pre-shared key (PSK) **conditions:** auth = psk
cert **values:** /storage certificate	Local certificate (including private key) **conditions:** auth = pubkey
leftsourceip **pattern example:** 192.168.1.1, %config	Virtual IP address for tunnel origination

rightsourceip **pattern example:** 192.168.1.1/24, 192.168.1.1-192.168.1.100, %config	Assign virtual IP to remote peer

13.3.3. profile

This section is used to define reusable **IKE Profiles**. Each profile is a named collection of cryptographic algorithms and parameters that govern the establishment of the Phase 1 security association (the IKE SA).

By creating these profiles, you can standardize and simplify your IPsec configuration. Instead of repeatedly defining the same set of algorithms for each connection, you can simply create a profile and reference it from multiple IKE *Connections*.

A profile specifies all the critical security parameters for the IKE negotiation, including:

- The allowed *Encryption*, *Hash*, and *Pseudo-random Function* algorithms.
- The *Diffie-Hellman (DH) Group* to be used for secure key exchange.
- The *IKE Lifetime*, which determines how long a Phase 1 SA remains valid before it must be re-negotiated.
- The Dead Peer Detection (*DPD*) parameters, which control how the router monitors the liveness of the IKE peer and what *Action* to take if the peer becomes unresponsive.

The *Apply* function in this section performs a soft reload, allowing for the addition or modification of policies without disrupting existing, active Security Associations. To manually terminate a specific SA, use the status page.

PROPERTIES

encryption **values:** aes128, aes192, aes256, 3des	Data encryption method for secure data transmission
hash **values:** sha1, sha256, sha384, sha512, md5	Integrity hash algorithm
prf **values:** sha1, sha256, sha384, sha512, md5	Pseudorandom function (PRF) for key derivation
dh **values:** modp768, modp1024, modp1536, modp2048, modp3072, modp4096, modp6144, modp8192	Diffie-Hellman key exchange group
ikelifetime	IKE/ISAKMP SA rekey interval (seconds)
dpddelay	Dead Peer Detection (DPD) packet sending interval
dpdtimeout	Terminate IKEv1 connection after a timeout without a response to the DPD packet
dpdaction **values:** hold, restart, clear, none	Action to perform when connection is broken due to timeout with no response to the DPD packet

13.3.4. proposal

This section is used to define reusable **IPsec Proposals**. Each proposal is a named collection of security algorithms that govern how the actual data traffic is protected within the IPsec tunnel (the Phase

2, or Child SA).

Similar to IKE Profiles, Proposals provide a modular way to manage security policies. By defining a proposal, you specify the set of *Encryption* and *Authentication* algorithms that will be used by the Encapsulating Security Payload (ESP) protocol to ensure data confidentiality and integrity.

Key parameters defined within a proposal

- The cryptographic algorithms for data protection (*Encryption*, *Auth*).
- The Perfect Forward Secrecy (*PFS*) Group, which ensures that a new, independent key is generated for each session, enhancing long-term security.
- The *Lifetime* of the Child SA, which determines how often the data encryption keys are re-negotiated.

These proposals are then referenced from the "Associations" section to easily apply standardized data protection policies across multiple tunnels.

PROPERTIES

encryption **values:** aes128, aes192, aes256, 3des	Data encryption method for secure data transmission
auth **values:** sha1, sha256, sha384, sha512, md5	Authentication algorithm
pfs **values:** modp768, modp1024, modp1536, modp2048, modp3072, modp4096, modp6144, modp8192, none	Perfect Forward Secrecy ensures that DH keys will not be reused
lifetime	Connection lifetime (sec) from negotiation to expiry

13.3.5. status

This section displays the real-time operational status of all configured IPsec tunnels. It serves as the primary diagnostic tool for verifying that the configured tunnels are active and for troubleshooting connectivity issues.

The page visualizes the outcome of the settings defined in the *Connections* and *Associations* sections. Here, an administrator can confirm that the correct peers (*LOCAL-ID* and *REMOTE-ID*) and traffic policies (*LOCAL-TS* and *REMOTE-TS*) have successfully negotiated and established a tunnel.

The *STATUS* column is particularly useful for diagnostics, as it provides specific information about the tunnel's state, such as *Waiting for traffic* or *Phase 1 established*, allowing for rapid identification of configuration or network problems. The table also provides essential operational data, including tunnel *UPTIME* and traffic counters (*BYTES-IN*/*BYTES-OUT*).

COMMANDS

debug	Change logging level level - Verbose logging level (required)

PROPERTIES

association	IPSec association **conditions:** association != ""
uniqueid	SA Security Parameter Index (SPI)

status	SA status
local	Tunnel source endpoint address
local-id	Local IKE identity string
remote	Tunnel destination endpoint address
remote-id	Remote IKE identity string
uptime	SA lifetime since establishment
encryption	Negotiated encryption algorithm
integrity	Negotiated integrity algorithm
prf	Negotiated pseudorandom function
dh	Negotiated Diffie-Hellman group
local-ts	Local traffic selector subnet
remote-ts	Remote traffic selector subnet
expires	SA remaining lifetime before rekey
reqid	SA unique identifier (SPI)
rx-tx	SA traffic counters (RX/TX)

13.4. l2tp

This section configures the router to act as an L2TPv2 client, enabling it to establish a secure tunnel to a remote L2TP server. The tunnel is established over the Point-to-Point Protocol (PPP) and is commonly used to connect to corporate networks or remote access services.

Configuration primarily involves specifying the remote server's *IP address* and providing the necessary *Username* and *Password* for authentication. The *Authentication type* can be selected to match the server's requirements.

Beyond basic connectivity, you can also enable *Encryption* (typically MPPE) to secure the data payload. The *Metric* field allows for control over the route's preference in the router's routing table, which is essential for multi-WAN or failover configurations.

PROPERTIES

disabled **values:** true, false	Disable configuration
local-ip **values:** /ip interface, /mobile modem, /tunnel eoip, /tunnel gre, /tunnel l2tp, /tunnel openvpn, /tunnel pptp, /tunnel wireguard interface, /tunnel l2tpv3, /tunnel atunnel, /defaults server, auto **pattern example:** bridge0, 192.168.1.1	Local endpoint address or interface
remote-ip **pattern example:** localhost, sample.example.com, 8.8.8.8	Remote endpoint (FQDN or IP address)
username	Client authentication username
password	Client authentication password
auth **values:** any, chap, mschap, mschap-v2, pap	Authentication protocol **conditions:** encryption = mppe
encryption **values:** none, mppe	Tunnel encryption method **conditions:** /defaults/ipsec_installed = true

psk **minLength:** 8	IPsec pre-shared key for tunnel authentication **conditions:** encryption = ipsec
ppp-option **values:** lcp-echo-failure, lcp-echo-interval, lcp-max-configure, lcp-max-failure, lcp-max-terminate, lcp-restart, ipcp-accept-local, ipcp-accept-remote, ipcp-max-configure, ipcp-max-failure, ipcp-max-terminate, ipcp-restart, mru, mtu	Point-to-Point Protocol Daemon (pppd) options
debug **values:** true, false	Verbose logging for detailed diagnostics

13.5. l2tp v3

This section configures L2TPv3 (Layer 2 Tunneling Protocol version 3) tunnels, an IETF-standardized protocol for creating point-to-point Layer 2 connections over an IP network.

Unlike its predecessor L2TPv2, L2TPv3 operates independently of PPP and is designed as a more versatile and streamlined L2 transport mechanism. Its primary function is to encapsulate and transport full Ethernet frames, effectively creating a "pseudo-wire" that bridges two remote network segments.

The configuration requires defining the tunnel endpoints (*Local IP*, *Remote IP*) and a set of unique identifiers (*Tunnel-Id*, *Session-Id*) that must match on both sides to establish the connection.

For security, L2TPv3 does not have its own native encryption. Instead, it is designed to be secured by an underlying transport-layer security protocol. This implementation supports enabling *IPsec* to encrypt the entire L2TPv3 tunnel.

PROPERTIES

disabled **values:** true, false	Disable configuration
local-ip **values:** /ip interface, /mobile modem, /tunnel eoip, /tunnel gre, /tunnel l2tp, /tunnel openvpn, /tunnel pptp, /tunnel wireguard interface, /tunnel l2tpv3, /tunnel atunnel, /defaults server, auto **pattern example:** bridge0, 192.168.1.1	Local endpoint address or interface
remote-ip **pattern example:** localhost, sample.example.com, 8.8.8.8	Remote endpoint (FQDN or IP address)
tunnel-ip **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1/255.255.255.0	Local tunnel interface IP address
encryption **values:** none	Tunnel encryption method **conditions:** /defaults/ipsec_installed = true
psk **minLength:** 8	IPsec pre-shared key for tunnel authentication **conditions:** encryption = ipsec
macaddr **pattern example:** FE:FF:FF:FF:FF:FF	MAC address for hardware identification
mtu **minimum:** 70, **maximum:** 65535	Interface MTU to configure maximum transmission unit size
l2spec-type **values:** default, none	Layer 2 header format specification

tunnel-id **minimum:** 1, **maximum:** 4294967295	Tunnel identifier
peer-tunnel-id **minimum:** 1, **maximum:** 4294967295	Remote tunnel identifier
session-id **minimum:** 1, **maximum:** 4294967295	Tunnel session identifier
peer-session-id **minimum:** 1, **maximum:** 4294967295	Remote session identifier
encap **values:** ip, udp	Protocol encapsulation mode
udp-dport **minimum:** 1024 **pattern example:** 80, !80	Remote peer UDP port for UDP encapsulated **conditions:** encap = udp
udp-sport **minimum:** 1024 **pattern example:** 80, !80	UDP port for UDP encapsulation **conditions:** encap = udp

13.6. openvpn

This section configures the router's OpenVPN client, a versatile and highly secure open-source VPN protocol. OpenVPN is widely used for creating both site-to-site and remote access VPNs by establishing an encrypted tunnel over a public network.

OpenVPN's security model is built upon the SSL/TLS protocol, using certificates and private keys for robust authentication and key exchange. Its flexibility allows it to operate over either *UDP* (preferred for performance) or *TCP* (for reliability and bypassing restrictive firewalls).

The configuration on this page allows you to define all aspects of the client connection:

- The type of virtual interface: *TUN* for a routed Layer 3 IP tunnel, or *TAP* for a bridged Layer 2 Ethernet tunnel.
- The remote server's address and the transport *protocol*.
- The required cryptographic parameters, including the *Cipher* and *Auth* algorithms.
- The necessary security credentials, such as the *CA* certificate, the client's *Certificate*, and its private key. For certain setups, *Username*/*Password* or a static *TLS-Auth* key can also be used.

COMMANDS

generate-ta-key	Generate a key file used as TA Key or Static Key filename - Configuration name (required)

PROPERTIES

disabled **values:** true, false	Disable configuration
local-ip **values:** /ip interface, /mobile modem, /tunnel eoip, /tunnel gre, /tunnel l2tp, /tunnel openvpn, /tunnel pptp, /tunnel wireguard interface, /tunnel l2tp-v3, /tunnel atunnel, /defaults server, auto **pattern example:** bridge0	Local endpoint address or interface
remote-ip	Remote endpoint (FQDN or IP address)

tunnel-ip **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1/255.255.255.0	Local tunnel interface IP address
dev-type **values:** tun, tap	Virtual interface type
protocol **values:** tcp, udp	Tunnel transport protocol selection
encryption **values:** none, tls, static-key	Data channel encryption method
static-key **values:** /storage file	Pre-shared key (PSK) file for encryption **conditions:** encryption = static-key
cipher **values:** AES-128-CBC, AES-128-GCM, AES-192-CBC, AES-192-GCM, AES-256-CBC, AES-256-GCM	Data channel encryption algorithm **conditions:** encryption = tls encryption = static-key
auth **values:** MD5, SHA1, SHA256, SHA384, SHA512, none	Authentication algorithm **conditions:** encryption = tls encryption = static-key
ta-key **values:** /storage file	Key for additional HMAC authentication **conditions:** encryption = tls
ca **values:** /storage certificate	Certificate Authority (CA) public key **conditions:** encryption = tls
cert **values:** /storage certificate	Local certificate (including private key) **conditions:** encryption = tls
username	Client authentication username **conditions:** encryption = tls
password	Client authentication password **conditions:** encryption = tls
flag **values:** allow-recursive-routing, auth-nocache, auth-user-pass-optional, client-to-client, comp-noadapt, fast-io, float, mtu-test, multihome, ncp-disable, nobind, opt-verify, passtos, persist-key, persist-local-ip, persist-remote-ip, persist-tun, ping-timer-rem, pull, push-peer-info, push-reset, remote-random, route-nopull, single-session, suppress-timestamps, tcp-nodelay, tls-client, tls-exit, tls-server, username-as-common-name	Tunnel advanced configuration flags
extra **values:** auth-retry, bcast-buffers, comp-lzo, compress, connect-freq, connect-retry, connect-retry-max, connect-timeout, ecdh-curve, explicit-exit-notify, fragment, hand-window, hash-size, ifconfig_local, ifconfig_remote, ifconfig_netmask, ifconfig-push-local, ifconfig-push-netmask, inactive, key-direction, keysize, link-mtu, lport, max-clients, mssfix, mtu-disc, ping, ping-exit, ping-restart, prng, pull-filter-accept, pull-filter-ignore, pull-filter-reject, rcvbuf, remote-cert-tls, reneg-bytes, reneg-pkts, reneg-sec, replay-persist, replay-window, resolv-retry, shaper, sndbuf, topology, tcp-queue-limit, tls-timeout, tls-version-min, tran-window, tun-mtu, tun-mtu-extra, txqueuelen, verb, verify-client-cert, verify-x509-name, x509-username-field	Tunnel extra parameters

13.7. pptp

This section configures the router to act as a PPTP (Point-to-Point Tunneling Protocol) client. This allows the router to establish a VPN tunnel to a remote PPTP server, typically for remote access to a private network.

PPTP is a widely supported but older VPN protocol. Its setup is straightforward, primarily requiring the *Remote IP* address of the server and the *Username* and *Password* for authentication.

The configuration page allows for the selection of an *Authentication type* (e.g., MS-CHAPv2) and enabling *Encryption* (MPPE) to provide a basic level of data confidentiality.

PPTP is a legacy protocol with known security vulnerabilities. Its use is strongly discouraged for any application where data security is a concern. For secure remote access, it is recommended to use modern VPN protocols such as IPsec or OpenVPN instead.

PROPERTIES

disabled **values:** true, false	Disable configuration
local-ip **values:** /ip interface, /mobile modem, /tunnel eoip, /tunnel gre, /tunnel l2tp, /tunnel openvpn, /tunnel pptp, /tunnel wireguard interface, /tunnel l2tp-v3, /tunnel atunnel, /defaults server, auto **pattern example:** bridge0	Local endpoint address or interface
remote-ip **pattern example:** localhost, sample.example.com, 8.8.8.8	Remote endpoint (FQDN or IP address)
username	Client authentication username
password	Client authentication password
auth **values:** any, chap, mschap, mschap-v2, pap, eap	Authentication protocol **conditions:** encryption = mppe
encryption **values:** none, mppe	Tunnel encryption method **conditions:** /defaults/ipsec_installed = true
psk **minLength:** 8	IPsec pre-shared key for tunnel authentication **conditions:** encryption = ipsec
ppp-option **values:** lcp-echo-failure, lcp-echo-interval, lcp-max-configure, lcp-max-failure, lcp-max-terminate, lcp-restart, ipcp-accept-local, ipcp-accept-remote, ipcp-max-configure, ipcp-max-failure, ipcp-max-terminate, ipcp-restart, mru, mtu	Point-to-Point Protocol Daemon (pppd) options
debug **values:** true, false	Verbose logging for detailed diagnostics

13.8. wireguard

13.8.1. interface

This section is used to configure the local WireGuard network interface. This interface acts as one endpoint of the secure tunnel, defined primarily by its cryptographic key pair. In WireGuard, the *Public*

Key serves as the sole identifier for the interface, replacing traditional usernames or complex certificates.

The core configuration involves generating a *Private Key* (from which the public key is derived) and assigning a unique *IP Address* within the VPN's virtual network. Additionally, the listening *Port* for incoming peer connections is defined here.

Configuring this interface is only the first step. To establish a functional tunnel, you must proceed to the *Peer* section to define the remote endpoints that are authorized to connect to this interface.

PROPERTIES

disabled **values:** true, false	Disable configuration
tunnel-ip **pattern example:** 192.168.1.1, 192.168.1.0/24, 192.168.1.1/255.255.255.0	Tunnel interface IP address
priv-key **pattern example:** ABC123abc456ABC789abc123ABC456abc789Abc123A=	Private key for authentication and traffic encryption
pub-key **pattern example:** ABC123abc456ABC789abc123ABC456abc789Abc123A=	Public key for authentication and traffic encryption
port **pattern example:** 22, 51820	Listening port for incoming connections
mtu **minimum:** 70, **maximum:** 65535	Interface MTU to configure maximum transmission unit size
fw-mark	The mark for packets associated with this interface

13.8.2. peer

This section is where the remote peers authorized to connect to the local WireGuard interface are defined. A peer represents a single remote endpoint in the VPN tunnel.

The configuration of a peer serves two critical, simultaneous functions: **authentication** and **routing**.

- **Authentication:** A peer is primarily identified by its *Public Key*. Only a peer that can prove ownership of the corresponding private key will be able to establish a connection. An optional *Preshared Key* can be added for an extra layer of symmetric-key security.
- **Routing:** The *Allowed IPs* list is a fundamental concept in WireGuard. It dictates which IP addresses are "owned" by this peer. Any traffic sent from the tunnel that originates from an IP address in this list will be accepted. Conversely, any traffic sent into the tunnel destined for an address in this list will be automatically encrypted and sent to this specific peer.

Optionally, a *Keepalive* interval can be configured to maintain persistent connections through stateful firewalls and NAT devices.

COMMANDS

generate-psk	Generate and apply PSK key

PROPERTIES

disabled **values:** true, false	Disable wireguard interface
interface **values:** /tunnel wireguard interface	WireGuard network interface
remote-ip **pattern example:** example.com:80, 192.168.1.1:80	Remote endpoint address
allowed-ip	Address associated with this peer to provide routing
pub-key **pattern example:** ABC123abc456ABC789abc123ABC456abc789Abc123A=	Public key for authentication and traffic encryption
psk **pattern example:** ABC123abc456ABC789abc123ABC456abc789Abc123A=	Additional key for increased security
keepalive **minimum:** 0, **maximum:** 65535	Keepalive packet transmission interval (sec)

14. wireless

14.1. adapter

This section provides for the low-level configuration of the physical Wi-Fi radio adapters installed in the router. These settings define the fundamental operational parameters of the hardware itself, directly controlling its radio frequency (RF) behavior and ensuring regulatory compliance.

Configuration at this layer involves setting country-specific operational constraints, selecting available channels, and defining transmission power. This is the initial setup step and a prerequisite for creating one or more virtual access points (VAPs) that will broadcast network SSIDs.

COMMANDS

scan	Search for WiFi networks

PROPERTIES

disabled **values:** true, false	Disable configuration
country **values:** AD, AE, AF, AI, AL, AM, AN, AR, AS, AT, AU, AW, AZ, BA, BB, BD, BE, BF, BG, BH, BL, BM, BN, BO, BR, BS, BT, BY, BZ, CA, CF, CH, CI, CL, CN, CO, CR, CU, CX, CY, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, ET, FI, FM, FR, GB, GD, GE, GF, GH, GL, GP, GR, GT, GU, GY, HK, HN, HR, HT, HU, ID, IE, IL, IN, IR, IS, IT, JM, JO, JP, KE, KH, KN, KP, KR, KW, KY, KZ, LB, LC, LI, LK, LS, LT, LU, LV, MA, MC, MD, ME, MF, MH, MK, MN, MO, MP, MQ, MR, MT, MU, MV, MW, MX, MY, NG, NI, NL, NO, NP, NZ, OM, PA, PE, PF, PG, PH, PK, PL, PM, PR, PT, PW, PY, QA, RE, RO, RS, RU, RW, SA, SE, SG, SI, SK, SN, SR, SV, SY, TC, TD, TG, TH, TN, TR, TT, TW, TZ, UA, UG, US, UY, UZ, VC, VE, VI, VN, VU, WF, WS, YE, YT, ZA, ZW, default	Selecting a country to apply regulatory restrictions
channels **values:** $_channels	List of wireless adapter channels

txpower **values:** auto, $_txpowers	Transmit power level
htmode **values:** none, $_htmodes	High Throughput (HT) mode
frag-threshold **minimum:** 256, **maximum:** 2346	The size (bytes) of a packet, above which it is fragmented. Even number
rts-threshold **minimum:** 0, **maximum:** 2347	The size (bytes) of a packet, above which the RTS/CTS mechanism is activated
beacon **minimum:** 15, **maximum:** 65535	AP beacon transmission interval (msec)

14.2. filter

This section is used to configure MAC address-based Access Control Lists (ACLs) for wireless networks. This feature provides an additional layer of security by restricting network access to only authorized devices.

The system implements a "whitelist" model. This means that when a filter is applied to a wireless network, only client devices whose MAC addresses are present in the list will be permitted to connect. All other devices will be denied access.

This page is for creating and managing the named filter lists themselves. The actual application of a filter to a specific wireless network is performed in the *Wireless / Network* section.

PROPERTIES

mac	Wi-Fi device MAC address

14.3. network

This section is where the logical wireless networks are defined, each of which operates on top of a physical *Adapter*. Each network configured here represents a distinct wireless service, such as an Access Point (AP) broadcasting an SSID, a client connecting to an upstream Wi-Fi network, or a node in a mesh.

Operational Modes

A key configuration choice is the operational *Mode*, which defines the role of this wireless network:

- **Access Point (ap):** The most common mode. The router broadcasts an *SSID* to which client devices can connect.
- **Station (sta):** Client mode. The router connects to another existing Wi-Fi network, typically to use it as a WAN uplink or to bridge networks.
- **Mesh Point (mesh):** A specialized mode for building a distributed, self-healing mesh network with other compatible devices.

Security and Access Control

Beyond defining the mode, this is where the essential security and access parameters for each network are configured:

- **Authentication:** Defining the security profile by selecting the *Encryption* type (e.g., WPA2/WPA3) and setting the network *Key*.
- **Access Control:** Applying a *MAC Filter* policy by selecting a pre-defined list created in the *Wireless / MAC Filter* section.
- **Client Isolation:** In *ap* mode, enabling *Isolate Clients* prevents wireless devices connected to the same SSID from communicating directly with each other.

PROPERTIES

disabled **values:** true, false	Disable configuration
adapter **values:** /wireless adapter	Wireless PHY (physical layer) adapter
mode **values:** ap, sta, mesh	Wireless network operation mode
encryption **values:** none, psk-mixed+ccmp, wpa-mixed+ccmp, sae	Wireless network encryption type **conditions:** mode = mesh
ssid	Wireless network SSID (Service Set Identifier)
key **minLength:** 8	Wireless network authentication password **conditions:** encryption = psk-mixed+ccmp encryption = sae
wds **values:** true, false	Use 4-address mode to allow transparent ethernet bridging **conditions:** mode != mesh
wmm **values:** true, false	Enable Wireless Multimedia (WMM) QoS for multimedia traffic
mesh-forward **values:** true, false	Enable forwarding of packets not destined for this mesh station **conditions:** mode = mesh
mesh-ttl	Maximum hop count for mesh network packets **conditions:** mode = mesh
mesh-rssi-threshold **minimum:** -100, **maximum:** 1	Minimum RSSI threshold for mesh network connectivity **conditions:** mode = mesh
hidden **values:** true, false	Disable beacon frame transmission **conditions:** mode = ap
mtu **minimum:** 256, **maximum:** 2304	Interface MTU to configure maximum transmission unit size
isolate **values:** true, false	Enable client-to-client communication blocking **conditions:** mode = ap
mac-filter **values:** deny, allow, disable	Wireless Access Control List (ACL) policy **conditions:** mode = ap && encryption != wpa-mixed+ccmp

mac-list **values:** /wireless filter	Wireless devices ACL **conditions:** mode = ap && mac-filter != disable && mac-filter != radius && encryption != wpa-mixed+ccmp
auth-server **pattern example:** 192.168.1.1, example.com, 192.168.1.1:80	Authentication server address **conditions:** mac-filter = radius encryption = wpa-mixed+ccmp && mode != sta
auth-secret	Authentication server shared key **conditions:** mac-filter = radius encryption = wpa-mixed+ccmp && mode != sta
eap-method **values:** peap, tls, ttls	EAP authentication method **conditions:** mode = sta && encryption = wpa-mixed+ccmp
eap-auth-method **values:** pap, mschap-v2	WPA/WPA2 enterprise authentication method **conditions:** encryption = wpa-mixed+ccmp && mode = sta && eap-protocol = ttls encryption = wpa-mixed+ccmp && mode = sta && eap-protocol = peap
eap-username **pattern example:** bridge0, vpn, client_1	WPA/WPA2 enterprise authentication username **conditions:** encryption = wpa-mixed+ccmp && mode = sta
eap-password	WPA/WPA2 enterprise authentication password **conditions:** encryption = wpa-mixed+ccmp && mode = sta
ca **values:** /storage certificate	CA certificate **conditions:** encryption = wpa-mixed+ccmp && mode = sta
cert **values:** /storage certificate	Client authentication certificate **conditions:** encryption = wpa-mixed+ccmp && mode = sta

15. Customer Support

Website	donyx.com
Email	support@donyx.com