Network configuration

The central network configuration is located in the file `/etc/config/network`. This configuration file is responsible for defining *switch VLANs*, *interface configurations* and *network routes*. After editing and saving `/etc/config/network` you need to execute

```
/etc/init.d/network reload
```

to stop and restart the network before any changes take effect. Rebooting the router is not necessary.

- https://dev.openwrt.org/browser/branches/attitude_adjustment/package/base-files/files/etc/config/network [https://dev.openwrt.org/browser/branches/attitude_adjustment/package/base-files/files/etc/config/network]
- https://dev.openwrt.org/browser/trunk/package/base-files/files/etc/config/network [https://dev.openwrt.org/browser/trunk/package/base-files/files/etc/config/network]

Feel free to inform yourself about netifd (Network Interface Daemon).

Sections

Below is an overview of the section types that may be defined in the network configuration. A minimal network configuration for a router usually consists of at least two *interfaces* (`lan` and `wan`) and a *switch* section if applicable.

Global Settings

The globals section is available in Barrier Breaker and later releases.

The `globals` section contains interface-independent options affecting the network configuration in general.

Name	Type	Required	Default	Description
`ula_prefix`	IPv6-prefix	no	*(none)*	IPv6 ULA [https://en.wikipedia.org/wiki/Unique local address]-Prefix for this device

Switch

The `switch` section is responsible for partitioning the switch into several *VLANs* which appear as independent interfaces in the system although they share the same hardware. **Not every OpenWrt supported device (or architecture, like x86) has a programmable switch**, therefore this section might not be present on some platforms. Please also note, that some switches only support 4Bit-VLANs.

There are currently two different configuration formats in use, one for the legacy `/proc/switch/` API and one for the newer `swconfig`-based switch configuration.

/proc/switch

This variant is actually only found on Broadcom devices like the WRT54GL.

A typical configuration for it looks like this:

```
config 'switch' 'eth0'
        option 'vlan0' '0 1 2 3 5*'
        option 'vlan1' '4 5'
```

The `eth0` identifier specifies the switch the section is belonging to. VLANs are defined by `vlan#` options with `#` being the VLAN number. For further information refer to the switch documentation.

swconfig

The newer `swconfig`-framework is intended to replace the legacy switch configuration.

Configuration for swconfig have a slightly different structure with one extra section per VLAN. The example below shows a typical configuration:

```
config 'switch' 'eth0'
        option 'reset' '1'
        option 'enable_vlan' '1'

config 'switch_vlan' 'eth0_1'
        option 'device' 'eth0'
        option 'vlan' '1'
        option 'ports' '0 1 2 3 5t'

config 'switch_vlan' 'eth0_2'
        option 'device' 'eth0'
        option 'vlan' '2'
        option 'ports' '4 5t'
```

Common properties are defined within the `switch` section; vlan specific properties are located in additional `switch_vlan` sections linked to the `switch` section through the `device` option. The complete layout is explained in the switch documentation.

Interfaces

Sections of the type `interface` declare logical networks serving as containers for IP address settings, aliases, routes, physical interface names and firewall rules - they play a central role within the OpenWrt configuration concept.

A minimal interface declaration consists of the following lines:

```
config 'interface' 'wan'
        option 'proto' 'dhcp'
        option 'ifname' 'eth0.1'
```

- `wan` is a unique *logical interface name*
- `dhcp` specifies the *interface protocol*, DHCP in this example
- `eth0.1` is the *physical interface* associated with this section

(!) The system limits the *physical interface name* length to 15 characters including the automatically added prefix that is added for some protocols (e.g. "6in4-", "pppoa-", "pppoe-") or due to bridge status ("br-"). Depending on the protocol type, the *logical interface name* may thus be limited to only 9 characters. E.g. 'abcde67890' is a valid interface name for a normal interface using dhcp, but not for a pppoe interface where the final name would be 'pppoe-abcde67890', which is >15 chars. Using a too long name may lead into errors, as some of the settings in network, firewall or dhcp config may be left unapplied.

The *interface protocol* may be one of the following:

Protocol	Description	Program
`static`	Static configuration with fixed address and netmask	`ip`/`ifconfig`
`dhcp`	Address and netmask are assigned by DHCP	`udhcpc` (Busybox)
`dhcpv6`	Address and netmask are assigned by DHCPv6	`odhcpc6c`
`ppp`	PPP protocol - dialup modem connections	`pppd`
`pppoe`	PPP over Ethernet - DSL broadband connection	`pppd` + `plugin rp-pppoe.so`
`pppoa`	PPP over ATM - DSL connection using a builtin modem	`pppd` + plugin …
`3g`	CDMA, UMTS or GPRS connection using an AT-style 3G modem	`comgt`
`qmi`	USB modems using QMI protocol	`uqmi`
`ncm`	USB modems using NCM protocol	`comgt-ncm` + ?
`wwan`	USB modems with protocol autodetection	`wwan`
`hnet`	Self-managing home network (HNCP)	`hnet-full`
`pptp`	Connection via PPtP VPN	?
`6in4`	IPv6-in-IPv4 tunnel for use with Tunnel Brokers like HE.net	?
`aiccu`	Anything-in-anything tunnel	`aiccu`
`6to4`	Stateless IPv6 over IPv4 transport	?
`6rd`	IPv6 rapid deployment	`6rd`
`dslite`	Dual-Stack Lite	`ds-lite`
`l2tp`	PPP over L2TP Pseudowire Tunnel	`xl2tpd`
`relay`	relayd pseudo-bridge	`relayd`
`gre`, `gretap`	GRE over IPv4	`gre` + `kmod-gre`
`grev6`, `grev6tap`	GRE over IPv6	`gre` + `kmod-gre6`
`vti`	VTI over IPv4	`vti` + `kmod-ip_vti`
`vtiv6`	VTI over IPv6	`vti` + `kmod-ip6_vti`
`none`	Unspecified protocol, therefore all the other interface settings will be ignored (like disabling the configuration)	-

Depending on the used *interface protocol* several other options may be required for a complete interface declaration. The corresponding options for each protocol are listed below. Options marked as "yes" in the "Required" column *must* be defined in the interface section if the corresponding protocol is used, options marked as "no" *may* be defined but can be omitted as well.

(!) In openwrt 12.09, if an interface section has no protocol defined (not even `none`), the other settings are completely ignored. The result is that, if the interface section is mentioning a physical network interface (i.e. eth0), this will be down even if a cable is connected (with proto 'none' the interface is up). (could be that more testing is needed)

Options valid for all protocol types

Name	Type	Required	Default	Description

Name	Type	Required	Default	Description
`ifname`	interface name(s)	yes(*)	*(none)*	Physical interface name to assign to this section, list of interfaces if type bridge is set. *(*) This option may be empty or missing if only a wireless interface references this network or if the protocol type is `pptp`, `pppoa` or `6in4`*
`type`	string	no	*(none)*	If set to "bridge", a bridge containing the given *ifnames* is created Wlan interface names are not predictable, therfore you cannot reference them directly in the network config [https://forum.openwrt.org/viewtopic.php?pid=203784#p203784]
`stp`	boolean	no	`0`	Only valid for type "bridge", enables the Spanning Tree Protocol
`bridge_empty`	boolean	no	`0`	Only valid for type "bridge", enables creating empty bridges
`igmp_snooping`	boolean	no	`1`	Only valid for type "bridge", sets the multicast_snooping kernel setting for a bridge
`macaddr`	mac address	no	*(none)*	Override MAC address of this interface
`mtu`	number	no	*(none)*	Override the default MTU on this interface
`auto`	boolean	no	`0` for proto `none`, else `1`	Specifies whether to bring up interface on boot
`ipv6`	boolean	no	`1`	Specifies whether to enable (1) or disable (0) IPv6 on this interface (Barrier Breaker and later only)
`accept_ra`	boolean	no	`1` for protocol `dhcp`, else `0`	**deprecated:** Specifies whether to accept IPv6 Router Advertisements on this interface (On Attitude Adjustment 12.09 and earlier versions)
`send_rs`	boolean	no	`1` for protocol `static`, else `0`	**deprecated:** Specifies whether to send Router Solicitations on this interface (On Attitude Adjustment 12.09 and earlier versions)
`force_link`	boolean	no	`1` for protocol `static`, else `0`	Specifies whether ip address, route, and optionally gateway are assigned to the interface regardless of the link being active ('1') or only after the link has become active ('0'); when set to '1', carrier sense events do not invoke hotplug handlers
`enabled`	boolean	no	`1`	enable or disable the interface section
`ip4table`	string	no	*(none)*	(ipv4) routing table for routes of this interface. E.g., when proto = dhcp, the dhcp client will add routes to that table
`ip6table`	string	no	*(none)*	(ipv6) routing table for routes of this interface. E.g., when proto = dhcp6, the dhcp6 client will add routes to that table

Protocol "static"

Name	Type	Required	Default	Description
`ipaddr`	ip address	yes, if no `ip6addr` is set.	*(none)*	IP address. [openwrt 12.09] It could be a list of ipaddr , that is: several ipaddresses will be assigned to the interface. If, instead of a list, several ipaddr are specified as options, only the last is applied.
`netmask`	netmask	yes, if no `ip6addr` is set	*(none)*	Netmask
`gateway`	ip address	no	*(none)*	Default gateway
`broadcast`	ip address	no	*(none)*	Broadcast address (autogenerated if not set)
`ip6addr`	ipv6 address	yes, if no `ipaddr` is set	*(none)*	Assign given IPv6 address to this interface (CIDR notation)
`ip6ifaceid`	ipv6 suffix	no	::1	Allowed values: 'eui64', 'random', fixed value like '::1:2'. When IPv6 prefix (like 'a:b:c:d::') is received from a delegating server, use the suffix (like '::1') to form the IPv6 address ('a:b:c:d::1') for this interface. Useful with several routers in LAN. The option was introduced by this commit [http://git.openwrt.org/?p=project/netifd.git;a=commitdiff;h=0b0e5e2fc5b065092644a5c4717c0a03a9098dcf;hp=e9d2014a478807c7fac0581bb4a145901a3f23b4] to netifd in Jan 2015.
`ip6gw`	ipv6 address	no	*(none)*	Assign given IPv6 default gateway to this interface
`ip6assign`	prefix length	no	*(none)*	Delegate a prefix of given length to this interface (Barrier Breaker and later only)
`ip6hint`	prefix hint (hex)	no	*(none)*	Hint the subprefix-ID that should be delegated as hexadecimal number (Barrier Breaker and later only)

Name	Type	Required	Default	Description
`ip6prefix`	ipv6 prefix	no	*(none)*	IPv6 prefix routed here for use on other interfaces (Barrier Breaker and later only)
`ip6class`	list of strings	no	*(none)*	Define the IPv6 prefix-classes this interface will accept
`dns`	list of ip addresses	no	*(none)*	DNS server(s)
`dns_search`	list of domain names	no	*(none)*	Search list for host-name lookup
`metric`	integer	no	`0`	Specifies the default route metric to use

Protocol "dhcp"

Name	Type	Required	Default	Description
~~`gateway`~~	~~string~~	~~no~~	~~*(none)*~~	~~Suppresses DHCP-assigned default gateway if set to 0.0.0.0~~ (deprecated)
`broadcast`	boolean	no	`0`	Enable the broadcast flag in DHCP requests, required for certain ISPs, e.g. Charter with DOCSIS 3
`ipaddr`	IP address	no	*(none)*	IP address to request from the DHCP server
`hostname`	string	no	*(none)*	Hostname to include in DHCP requests
`clientid`	string	no	*system default*	Override client identifier in DHCP requests
`vendorid`	string	no	*system default*	Override the vendor class in DHCP requests
`dns`	list of ip addresses	no	*(none)*	Supplement DHCP-assigned DNS server(s), or use only these if peerdns is 0
`peerdns`	boolean	no	`1`	Use DHCP-provided DNS server(s)
`defaultroute`	boolean	no	`1`	Whether to create a default route via the received gateway
`customroutes`	string	no	*(none)*	Space-separated list of additional routes to insert via the received gateway
`metric`	integer	no	`0`	Specifies the route metric to use for both default route and custom routes
`classlessroute`	boolean	no	`1`	Whether to request the "classless route" option (DHCP option 121) — *available since LEDE r2001*
`reqopts`	string	no	*(none)*	Space-separated list of additional DHCP options to request from the server
`sendopts`	string	no	*(none)*	Space-separated list of additional DHCP options to send to the server. Syntax: `option:value` where `option` is either an integer code or a symbolic name such as `hostname`.
`zone`	firewall zone	no	*(none)*	Firewall zone to which this interface should be added
`iface6rd`	logical interface	no	*(none)*	Logical interface template for auto-configuration of 6rd
`mtu6rd`	integer	no	*system default*	MTU of the 6rd interface
`zone6rd`	firewall zone	no	*system default*	Firewall zone to which the 6rd interface should be added

Note: To automatically configure 6rd from dhcp you need to create an interface with `option auto 0` and put its name as the 'iface6rd' parameter. In addition you also need to add its name to a suitable firewall zone in /etc/config/firewall.

(!) These parameters are handled partially by netifd (in `interface.c`) and partially by a shell script in `lib/netifd/proto/dhcp.sh`.

(!) It seems that if an interface is configured as dhcp client, at least on OpenWrt 10.03, the default route received by dhcp will be the only one listed and will remove other default route/metrics defined for other interfaces if those interfaces comes "before" the interface with dhcp in terms of "ifname" values. For example:

```
config interface wan
    option ifname eth0
    option proto static
    ..other options..

config interface wan2
    option ifname eth1
    option proto dhcp
    ..other options..
```

The interface with dhcp comes after (because eth1 comes after eth0 in a lexicografical order) and will overwrite the default routes set up by the interface "wan". While is not true the contrary. If we have:

```
config interface wan
    option ifname eth0
    option proto dhcp
    ..other options..

config interface wan2
    option ifname eth1
    option proto static
    ..other options..
```

Both default routes set up by wan and wan2 will appear in the routing table.

Protocol "dhcpv6"

The package `odhcp6c` must be installed to use dhcpv6.

Name	Type	Required	Default	Description
`reqaddress`	[try,force,none]	no	try	Behaviour for requesting addresses
`reqprefix`	[auto,no,0-64]	no	auto	Behaviour for requesting prefixes (numbers denote hinted prefix length). Use 'no' if you only want a single IPv6 address for the AP itself without a subnet for routing
`clientid`	hexstring	no	*system default*	Override client identifier in DHCP requests
`ifaceid`	ipv6 addr	no	*link-local identifier*	Override the interface identifier for adresses received via RA
`dns`	list of ip addresses	no	*(none)*	Supplement DHCP-assigned DNS server(s), or use only these if peerdns is 0
`peerdns`	boolean	no	`1`	Use DHCP-provided DNS server(s)
`defaultroute`	boolean	no	`1`	Whether to create an IPv6 default route via the received gateway
`reqopts`	list of numbers	no	*(none)*	Specifies a list of additional DHCP options to request
`noslaaconly`	boolean	no	`0`	Don't allow configuration via SLAAC (RAs) only (implied by reqprefix != no)
`norelease`	boolean	no	`0`	Don't send a RELEASE when the interface is brought down
`ip6prefix`	ipv6 prefix	no	*(none)*	Use an (additional) user-provided IPv6 prefix for distribution to clients
`iface_dslite`	logical interface	no	*(none)*	Logical interface template for auto-configuration of DS-Lite

Note: To automatically configure ds-lite from dhcpv6 you need to create an interface with `option auto 0` and put its name as the 'iface_dslite' parameter. In addition you also need to add its name to a suitable firewall zone in /etc/config/firewall.

Protocol "ppp" (PPP over Modem)

The package `ppp` must be installed to use PPP.

Name	Type	Required	Default	Description
`device`	file path	yes	*(none)*	Modem device node
`username`	string	no(?)	*(none)*	Username for PAP/CHAP authentication
`password`	string	no(?)	*(none)*	Password for PAP/CHAP authentication
`connect`	file path	no	*(none)*	Path to custom PPP connect script
`disconnect`	file path	no	*(none)*	Path to custom PPP disconnect script
`keepalive`	number	no	*(none)*	Number of unanswered echo requests before considering the peer dead. The interval between echo requests is 5 seconds.
`demand`	number	no	*(none)*	Number of seconds to wait before closing the connection due to inactivity
`defaultroute`	boolean	no	`1`	Replace existing default route on PPP connect
`peerdns`	boolean	no	`1`	Use peer-assigned DNS server(s)
`dns`	list of ip addresses	no	*(none)*	Override peer-assigned DNS server(s)
`ipv6`	boolean	no	`0`	Enable IPv6 on the PPP link
`pppd_options`	string	no	*(none)*	Additional command line arguments to pass to the pppd daemon

Protocol "pppoe" (PPP over Ethernet)

The packages `ppp`, `kmod-pppoe` and `ppp-mod-pppoe` must be installed to use PPPoE.

```
opkg update
opkg install ppp kmod-pppoe ppp-mod-pppoe
```

Name	Type	Required	Default	Description
`username`	string	no(?)	*(none)*	Username for PAP/CHAP authentication
`password`	string	no(?)	*(none)*	Password for PAP/CHAP authentication
`ac`	string	no	*(none)*	Specifies the Access Concentrator to connect to. If unset, `pppd` uses the first discovered one
`service`	string	no	*(none)*	Specifies the Service Name to connect to, If unset, `pppd` uses the first discovered one
`host_uniq`	string	no	*(none)*	Populate the Host-Uniq tag with this value, required by some ISPs
`connect`	file path	no	*(none)*	Path to custom PPP connect script
`disconnect`	file path	no	*(none)*	Path to custom PPP disconnect script
`keepalive`	number	no	*(none)*	Number of connection failures before reconnect
`demand`	number	no	*(none)*	Number of seconds to wait before closing the connection due to inactivity
`defaultroute`	boolean	no	`1`	Replace existing default route on PPP connect
`peerdns`	boolean	no	`1`	Use peer-assigned DNS server(s)
`dns`	list of ip addresses	no	*(none)*	Override peer-assigned DNS server(s)
`ipv6`	boolean	no	`0`	Enable IPv6 on the PPP link
`pppd_options`	string	no	*(none)*	Additional command line arguments to pass to the pppd daemon, e.g. **debug**

Protocol "pppoa" (PPP over ATM AAL5)

The package `ppp-mod-pppoa` must be installed to use PPPoA.

Name	Type	Required	Default	Description
`vci`	number	no	`35`	PPPoA VCI
`vpi`	number	no	`8`	PPPoA VPI
`atmdev`	number	no	`0`	Specifies the ATM adapter number starting with 0. Most systems only have one ATM device and do not need this option
`encaps`	string	no	`llc`	PPPoA encapsulation mode: 'llc' (LLC) or 'vc' (VC)
`username`	string	no(?)	*(none)*	Username for PAP/CHAP authentication
`password`	string	no(?)	*(none)*	Password for PAP/CHAP authentication
`connect`	file path	no	*(none)*	Path to custom PPP connect script
`disconnect`	file path	no	*(none)*	Path to custom PPP disconnect script
`keepalive`	number	no	*(none)*	Number of connection failures before reconnect
`demand`	number	no	*(none)*	Number of seconds to wait before closing the connection due to inactivity
`defaultroute`	boolean	no	`1`	Replace existing default route on PPP connect
`peerdns`	boolean	no	`1`	Use peer-assigned DNS server(s)
`dns`	list of ip addresses	no	*(none)*	Override peer-assigned DNS server(s)
`ipv6`	boolean	no	`0`	Enable IPv6 on the PPP link
`pppd_options`	string	no	*(none)*	Additional command line arguments to pass to the pppd daemon

Protocol "3g" (PPP over EV-DO, CDMA, UMTS or GPRS)

The package `comgt` must be installed to use 3G. Check 3gdongle for further help with that.

Name	Type	Required	Default	Description
`device`	file path	yes	*(none)*	Modem device node
`service`	string	yes	`umts`	3G service type: `cdma`/`evdo`, `umts`/`umts_only`/`gprs_only` (…._only options limited to Novatel & Option cards and dongles)
`apn`	string	yes	*(none)*	Used APN
`pincode`	number	no	*(none)*	PIN code to unlock SIM card
`dialnumber`	string	no	*99***1#	Modem dial string e.g. *99#
`maxwait`	number	no	`20`	Number of seconds to wait for modem to become ready

Name	Type	Required	Default	Description
`username`	string	no(?)	*(none)*	Username for PAP/CHAP authentication
`password`	string	no(?)	*(none)*	Password for PAP/CHAP authentication
`keepalive`	number	no	*(none)*	Number of connection failures before reconnect
`demand`	number	no	*(none)*	Number of seconds to wait before closing the connection due to inactivity
`defaultroute`	boolean	no	`1`	Replace existing default route on PPP connect
`peerdns`	boolean	no	`1`	Use peer-assigned DNS server(s)
`dns`	list of ip addresses	no	*(none)*	Override peer-assigned DNS server(s)
`ipv6`	boolean	no	`0`	Enable IPv6 on the PPP link

Protocol "qmi" (USB modems using QMI protocol)

The package `uqmi` must be installed to use QMI.

Name	Type	Required	Default	Description
`device`	file path	yes	*(none)*	QMI device node, typically /dev/cdc-wdm0
`apn`	string	yes	*(none)*	Used APN
`pincode`	number	no	*(none)*	PIN code to unlock SIM card
`username`	string	no	*(none)*	Username for PAP/CHAP authentication
`password`	string	no	*(none)*	Password for PAP/CHAP authentication
`auth`	string	no	*(none)*	Authentication type: pap, chap, both, none
`modes`	string	no	*(modem default)*	Allowed network modes, comma separated list of: all, lte, umts, gsm, cdma, td-scdma
`delay`	number	no	0	Seconds to wait before trying to interact with the modem (some ZTE modems require up to 30 s.)

Protocol "ncm" (USB modems using NCM protocol)

The package `comgt-ncm` + modem specific driver must be installed to use NCM.

Name	Type	Required	Default	Description
`device`	file path	yes	*(none)*	NCM device node, typically /dev/cdc-wdm0 or /dev/ttyUSB#
`apn`	string	yes	*(none)*	Used APN
`pincode`	number	no	*(none)*	PIN code to unlock SIM card
`username`	string	no	*(none)*	Username for PAP/CHAP authentication
`password`	string	no	*(none)*	Password for PAP/CHAP authentication
`auth`	string	no	*(none)*	Authentication type: pap, chap, both, none
`mode`	string	no	*(modem default)*	Used network mode, not every device support every mode: preferlte, preferumts, lte, umts, gsm, auto
`pdptype`	string	no	`IPV4V6`	Used IP-stack mode, `IP` (for IPv4), `IPV6` (for IPv6) or `IPV4V6` (for dual-stack) (Designated Driver #46844 and later)
`delay`	number	no	0	Seconds to wait before trying to interact with the modem (some modems require up to 30 s.)

Protocol "wwan" (USB modems autodetecting above protocols)

The package `wwan` must be installed to use this feature. The "wwan" protocol detects the right protocol (3G/QMI/NCM/MBIM) for the USB Modem model and passes the configuration to the protocol.

Name	Type	Required	Default	Description
`apn`	string	yes	*(none)*	Used APN
`auth`	string	no	*(none)*	Authentication type: pap, chap, both, none
`username`	string	no	*(none)*	Username for PAP/CHAP authentication
`password`	string	no	*(none)*	Password for PAP/CHAP authentication
`pincode`	number	no	*(none)*	PIN code to unlock SIM card
`modes`	string	no	*(modem default)*	Allowed network modes, comma separated list of: all, lte, umts, gsm, cdma, td-scdma
`delay`	number	no	0	Seconds to wait before trying to interact with the modem (some ZTE modems require up to 30 s.)

Protocol "hnet" (Self-managing home network (HNCP))

The package `hnet-full` must be installed to use hnet. We recommend you install `ipset` before installing `hnet-full` See https://tools.ietf.org/html/rfc7788 [https://tools.ietf.org/html/rfc7788] for details.

Name	Type	Required	Default	Description
`mode`	string	no	auto	Interface mode. One of external, guest, adhoc or hybrid.
`ip6assign`	integer	no	64	IPv6-prefix size to assign to this interface if internal.
`ip4assign`	integer	no	24	IPv4-prefix size to assign to this interface if internal.
`dnsname`	string	no	<device-name>	DNS-Label to assign to interface.

Protocol "pptp" (Point-to-Point Tunneling Protocol)

The package `pptp` must be installed to use PPtP. There is a separate Howto for this: vpn.client.pptp. You need to have another section to configure the "parent" device, and you might need to add "<vpn>" to your "wan" zone in the firewall (<vpn> being the "logical interface name" of this section).

Name	Type	Required	Default	Description
`server`	ip address	yes	*(none)*	Remote PPtP server
`username`	string	no(?)	*(none)*	Username for PAP/CHAP authentication
`password`	string	no(?)	*(none)*	Password for PAP/CHAP authentication
`buffering`	boolean	no	`1`	~~Enables buffering and reordering of packets, `0` disables it (`-nobuffer`)~~ pptp buffering option removed in r32482
`keepalive`	integer	no	?	Number of attempts to reconnect
`defaultroute`	boolean	no	`1`	Whether to create a default route over the tunnel
`peerdns`	boolean	no	`1`	Use PPTP-provided DNS server(s)
`delegate`	boolean	no	?	Use builtin IPv6-management
`iface`	string	no(?)	`pptp-<vpn>`	Name of the physical interface. Defaults to `pptp-<vpn>` no matter what you use

Protocol "6in4" (IPv6-in-IPv4 Tunnel)

The package `6in4` must be installed to use this protocol.

Name	Type	Required	Default	Description
`ipaddr`	IPv4 address	no	Current WAN IPv4 address	Local IPv4 endpoint address
`peeraddr`	IPv4 address	yes	*(none)*	Remote IPv4 endpoint address
`ip6addr`	IPv6 address (CIDR)	yes	*(none)*	Local IPv6 address delegated to the tunnel endpoint
`ip6prefix`	IPv6 prefix	no	*(none)*	Routed IPv6 prefix for downstream interfaces (Barrier Breaker and later only)
`sourcerouting`	boolean	no	`1`	Whether to route only packets from delegated prefixes (Barrier Breaker and later only)
`defaultroute`	boolean	no	`1`	Whether to create an IPv6 default route over the tunnel
`ttl`	integer	no	`64`	TTL used for the tunnel interface
`tos`	string	no	*(none)*	Type Of Service : either "inherit" (the outer header inherits the value of the inner header) or an hexadecimal value. Also known as DSCP. (Chaos Calmer and later only)
`mtu`	integer	no	`1280`	MTU used for the tunnel interface
`tunnelid`	integer	no	*(none)*	HE.net global tunnel ID (used for endpoint update)
`username`	string	no	*(none)*	HE.net username which you use to login into tunnelbroker, not the User ID shows after you have login int (used for endpoint update)
`password`	string	no	*(none)*	~~md5sum of~~ HE.net password (used for endpoint update)
`updatekey`	string	no	*(none)*	HE.net updatekey, overrides password (used for endpoint update)
`metric`	integer	no	`0`	Specifies the default route metric to use

Note: This protocol type does not need an `ifname` option set in the interface section. The interface name is derived from the section name, e.g. `config interface sixbone` would result in an interface named `6in4-sixbone`.

Note: HE.net has introduced updatekey as default for new tunnels in February 2014. Support added to Openwrt trunk by r39646.

Note: as of r41358 **username**, **password** and **updatekey** are all plaintext entries.

Note: although ip6prefix isn't required, sourcerouting, enabled by default, will prevent forwarding of packets unless ip6prefix is specified.

Protocol "aiccu" (Automatic IPv6 Connectivity Client Utility)

The package `aiccu` must be installed to use this protocol. This utility is not meant to be operated in a headless mode. Do not use it if you have some other option. Only AYIYA tunnel type has been tested. For static or heartbeat tunnels, use native 6in4 tunnel instead, perhaps with the he.net Tunnel Broker.

This protocol is available for Barrier Breaker and newer versions only.

Name	Type	Required	Default	Description
`username`	string	yes	*(none)*	Server username
`password`	string	yes	*(none)*	Server password
`protocol`	string	no	*(none)*	Tunnel setup protocol to use (`tic`, `tsp`, `l2tp`)
`server`	string	no	`tic.sixxs.net`	Tunnel setup server to use
`ip6addr`	IPv6 address (CIDR)	no	*(none)*	Local IPv6 address delegated to the tunnel endpoint (not necessary)
`ntpsynctimeout`	integer	no	`90`	Wait for NTP sync that many seconds (available since aiccu 20070115-12 [https://github.com/openwrt/packages/pull/579])
`tunnelid`	integer	no	*(none)*	TIC server tunnel ID
`ip6prefix`	IPv6 prefix	no	*(none)*	Routed IPv6 prefix for downstream interfaces
`defaultroute`	boolean	no	`1`	Whether to create an IPv6 default route over the tunnel
`sourcerouting`	boolean	no	`1`	Whether to route only packets from delegated prefixes
`tunnelid`	integer	no	*(none)*	TIC server tunnel ID
`requiretls`	boolean	no	`0`	Require TLS connection to TIC server
`nat`	boolean	no	`1`	Notify the user that a NAT-kind network is detected
`heartbeat`	boolean	no	`1`	Make heartbeats
`verbose`	boolean	no	`0`	Verbose logging to system log

Note: This protocol type does not need an `ifname` option set in the interface section. The interface name is derived from the section name, e.g. `config interface sixbone` would result in an interface named `aiccu-sixbone`.

Protocol "6to4" (IPv6-in-IPv4 Tunnel)

The package `6to4` must be installed to use this protocol.

Name	Type	Required	Default	Description
`ipaddr`	IPv4 address	no	Current WAN IPv4 address	Local IPv4 endpoint address
`defaultroute`	boolean	no	`1`	Whether to create an IPv6 default route over the tunnel
`ttl`	integer	no	`64`	TTL used for the tunnel interface
`tos`	string	no	*(none)*	Type Of Service : either "inherit" (the outer header inherits the value of the inner header) or an hexadecimal value (Chaos Calmer and later only)
`mtu`	integer	no	`1280`	MTU used for the tunnel interface
`metric`	integer	no	`0`	Specifies the default route metric to use
`adv_interface`	string	no	`lan`	(deprecated) The *logical interface name* of the network the subnet should be advertised on. Multiple interface names can be given.
`adv_subnet`	hex number	no	`1`	(deprecated) A subnet ID between `1` and `FFFF` which selects the advertised /64 prefix from the mapped 6to4 space. The subnet ID is incremented by 1 for every interface specified in `adv_interface`.
`adv_valid_lifetime`	integer	no	`300`	(deprecated) Overrides the advertised valid prefix lifetime, in seconds (see also radvd prefix options)
`adv_preferred_lifetime`	integer	no	`120`	(deprecated) Overrides the advertised preferred prefix lifetime, in seconds (see also radvd prefix options)

Note: This protocol type does not need an `ifname` option set in the interface section. The interface name is derived from the section name, e.g. `config interface wan6` would result in an interface named `6to4-wan6`.
Note: If radvd is installed and enabled, the 6to4 scripts will add a temporary prefix and interface declaration to the *radvd* uci configuration and perform a daemon restart if required. (deprecated)

Protocol "6rd" (IPv6 rapid deployment)

The package `6rd` must be installed to use this protocol.

The needed tunnel values are usually obtained via the DHCPv4 request for the WAN interface. Try that first. Below is only needed for hardcoding the tunnel.

Name	Type	Required	Default	Description

Name	Type	Required	Default	Description
`peeraddr`	IPv4 address	yes	no	6rd - Gateway
`ipaddr`	IPv4 address	no	Current WAN IPv4 address	Local IPv4 endpoint address
`ip6prefix`	IPv6 prefix (without length)	yes	no	6rd-IPv6 Prefix
`ip6prefixlen`	IPv6 prefix length	yes	no	6rd-IPv6 Prefix length
`ip4prefixlen`	IPv6 prefix length	no	0	IPv4 common prefix
`defaultroute`	boolean	no	`1`	Whether to create an IPv6 default route over the tunnel
`ttl`	integer	no	`64`	TTL used for the tunnel interface
`tos`	string	no	*(none)*	Type Of Service : either "inherit" (the outer header inherits the value of the inner header) or an hexadecimal value (Chaos Calmer and later only)
`mtu`	integer	no	`1280`	MTU used for the tunnel interface

Note: This protocol type does not need an `ifname` option set in the interface section. The interface name is derived from the section name, e.g. `config interface wan6` would result in an interface named `6rd-wan6`.

Note: Some ISP's give you the number of bytes you should use from your WAN IP to calculate your IPv6 address. ip4prefixlen expects the *prefix* bytes of your WAN IP to calculate the IPv6 address. So if your ISP gives you 14 bytes to calculate, enter 18 (32 - 14).

Protocol "dslite" (Dual-Stack Lite)

(!) The package `ds-lite` must be installed to use this protocol.

Name	Type	Required	Default	Description
`peeraddr`	IPv6 address	yes	no	DS-Lite AFTR address
`ip6addr`	IPv6 address	no	Current WAN IPv6 address	Local IPv6 endpoint address
`tunlink`	Logical Interface	no	Current WAN interface	Tunnel base interface
`defaultroute`	boolean	no	`1`	Whether to create an IPv6 default route over the tunnel
`ttl`	integer	no	`64`	TTL used for the tunnel interface
`mtu`	integer	no	`1280`	MTU used for the tunnel interface

(!) ds-lite operation requires that IPv4 NAT is disabled. You should adjust your settings in /etc/config/firewall accordingly.

Note: This protocol type does not need an `ifname` option set in the interface section. The interface name is derived from the section name, e.g. `config interface wan` would result in an interface named `dslite-wan`.

Protocol "l2tp" (PPP over L2TP Pseudowire Tunnel)

(!) The package `xl2tpd` must be installed to use this protocol.

Most options are similar to protocol "ppp".

Name	Type	Required	Default	Description
`server`	string	yes	*(none)*	L2TP server to connect to. Acceptable datatypes are hostname or IP address, with optional port separated by colon `:`. Note that specifying port is only supported recently and should appear in DD release
`username`	string	no	*(none)*	Username for PAP/CHAP authentication
`password`	string	yes if `username` is provided	*(none)*	Password for PAP/CHAP authentication
`ipv6`	bool	no	0	Enable IPv6 on the PPP link (IPv6CP)
`mtu`	int	no	`pppd` default	Maximum Transmit/Receive Unit, in bytes
`keepalive`	string	no	*(none)*	Number of unanswered echo requests before considering the peer dead. The interval between echo requests is 5 seconds.
`checkup_interval`	int	no	*(none)*	Number of seconds to pass before checking if the interface is not up since the last setup attempt and retry the connection otherwise. Set it to a value sufficient for a successful L2TP connection for you. It's mainly for the case that netifd sent the connect request yet xl2tpd failed to complete it without the notice of netifd
`pppd_options`	string	no	*(none)*	Additional options to pass to `pppd`

The name of the physical interface will be "l2tp-<logical interface name>".

Protocol "relay" (Relayd Pseudo Bridge)

(!) The package `relayd` must be installed to use this protocol.

Name	Type	Required	Default	Description
`network`	list of *logical interface names*	yes	*(none)*	Specifies the networks between which traffic is relayed
`gateway`	IPv4 address	no	*(network default)*	Override the gateway address sent to clients within DHCP responses
`expiry`	integer	no	`30`	Host expiry timeout in seconds
`retry`	integer	no	`5`	Number of ARP ping retries before a host is considered dead
`table`	integer	no	`16800`	Table ID for automatically added routes
`forward_bcast`	boolean	no	`1`	Enables forwarding of broadcast traffic, `0` disables it
`forward_dhcp`	boolean	no	`1`	Enables forwarding of DHCP requests and responses, `0` disables it

Common options for GRE protocols

(!) The package `gre` must be installed to use GRE. Additionally, you need `kmod-gre` and/or `kmod-gre6`.

GRE support has been introduced in Barrier Breaker. Four protocols are defined: "gre", "gretap", "grev6", and "grev6tap". The name of the GRE interface will be `gre-<logical interface name>` for "gre" and "gretap", and `grev6-<logical interface name>` for "grev6" and "grev6tap".

All four protocols accept the following common options:

Name	Type	Required	Default	Description
`mtu`	integer	no	1280	MTU
`ttl`	integer	no	64	TTL of the encapsulating packets
`tunlink`	logical interface name	no	*(none)*	Bind the tunnel to this interface (`dev` option of "ip tunnel")
`zone`	zone name	no	"wan"	Firewall zone to which the interface will be added
`tos`	string	no	*(none)*	Type of Service (IPv4), Traffic Class (IPv6): either "inherit" (the outer header inherits the value of the inner header) or an hexadecimal value (Chaos Calmer and later only)
`ikey`	integer	no	0	key for incoming packets
`okey`	integer	no	0	key for outgoing packets
`icsum`	boolean	no	false	require incoming checksum
`ocsum`	boolean	no	false	compute outgoing checksum
`iseqno`	boolean	no	false	require incoming packets serialisation
`oseqno`	boolean	no	false	perform outgoing packets serialisation

Protocol "gre" (GRE tunnel over IPv4)

The following options are supported, in addition to all common options above:

Name	Type	Required	Default	Description
`ipaddr`	IPv4 address	no	WAN IP	Local endpoint
`peeraddr`	IPv4 address	yes	*(none)*	Remote endpoint
`df`	boolean	no	true	Set "Don't Fragment" flag on encapsulating packets

Protocol "gretap" (Ethernet GRE tunnel over IPv4)

The following options are supported, in addition to all common options above:

Name	Type	Required	Default	Description
`ipaddr`	IPv4 address	no	WAN IP	Local endpoint
`peeraddr`	IPv4 address	yes	*(none)*	Remote endpoint
`df`	boolean	no	true	Set "Don't Fragment" flag on encapsulating packets
`network`	logical interface name	no	*(none)*	Logical network to which the tunnel will be added (bridged)

Protocol "grev6" (GRE tunnel over IPv6)

The following options are supported, in addition to all common options above:

Name	Type	Required	Default	Description
ip6addr	IPv6 address	no	WAN IP	Local endpoint
peer6addr	IPv6 address	yes	*(none)*	Remote endpoint
weakif	logical interface name	no	lan	Logical network from which to select the local endpoint if ip6addr parameter is empty and no WAN IP is available

Protocol "grev6tap" (Ethernet GRE tunnel over IPv6)

The following options are supported, in addition to all common options above:

Name	Type	Required	Default	Description
ip6addr	IPv6 address	no	WAN IP	Local endpoint
peer6addr	IPv6 address	yes	*(none)*	Remote endpoint
weakif	logical interface name	no	lan	Logical network from which to select the local endpoint if ip6addr is empty and no WAN IP is available
network	logical interface name	no	*(none)*	Logical network to which the tunnel will be added (bridged)

Protocol "vti" (VTI tunnel over IPv4)

VTI Tunnels are IPsec policies with a fwmark set. The traffic is redirected to the matching VTI interface.

Name	Type	Required	Default	Description
ipaddr	IPv4 address	no	WAN IP	Local endpoint
peeraddr	IPv4 address	yes	*(none)*	Remote endpoint
mtu	integer	no	1280	MTU
tunlink	logical interface name	no	*(none)*	Bind the tunnel to this interface (dev option of "ip tunnel")
zone	zone name	no	"wan"	Firewall zone to which the interface will be added
ikey	integer	no	0	key/fwmark for incoming packets
okey	integer	no	0	key/fwmark for outgoing packets

Protocol "vtiv6" (VTI tunnel over IPv6)

The following options are supported, in addition to all common options above:

Name	Type	Required	Default	Description
ip6addr	IPv6 address	no	WAN IP	Local endpoint
peer6addr	IPv6 address	yes	*(none)*	Remote endpoint
mtu	integer	no	1280	MTU
tunlink	logical interface name	no	*(none)*	Bind the tunnel to this interface (dev option of "ip tunnel")
zone	zone name	no	"wan"	Firewall zone to which the interface will be added
ikey	integer	no	0	key/fwmark for incoming packets
okey	integer	no	0	key/fwmark for outgoing packets

Devices

A minimal device declaration consists of the following lines:

```
config device 'eth0.106'
        option type '8021q'
        option name 'eth0.106'
        option ifname 'eth0'
        option vid '106'
```

VLAN Interfaces

VLAN Interfaces may be configured also. If not, they are created on the fly by netifd. Defining VLANs gives more options. The following options are supported:

Name	Type	Required	Default	Description
type	VLAN Type	no	802.1q	VLAN type, possible values: 8021q or 8021ad
name	Name	yes	*(none)*	Name of device, i.e. eth0.5 or vlan5

Name	Type	Required	Default	Description
`ifname`	Parent interface	yes	*(none)*	Name of parent/base interface, i.e. eth0
`vid`	VLAN Id	yes	*(none)*	VLAN Id
`macaddr`	MAC	no	*(none)*	MAC of new interface

MAC address option is send upstream but not merged at time of writng.

ATM Bridges (Ethernet over ATM AAL5)

(!) The package `br2684ctl` must be installed to use Ethernet over AAL5.

ATM bridges use a special config section called `atm-bridge`. Each `atm-bridge` section maps the specified ATM curcuit an `atm#` pseudo ethernet device which can be used for example in conjunction with `pppoe` to establish a DSL connection to the ISP.

A typical bridge section looks like this:

```
config atm-bridge
       option unit     '0'
       option vpi      '8'
       option vci      '35'
```

- Unit `0` will let `br2684ctl` create a `nas0` pseudo device
- VPI `8` and VCI `35` specifies the circuit to bridge. Those values are ISP dependant.

The `atm-bridge` section allows the following options:

Name	Type	Required	Default	Description
`unit`	number	yes	`0`	Specifies the br2684 interface number. If ommitted, `0` is assumed which would result in a `nas0` pseudo interface.
`vci`	number	no	`35`	PPPoA VCI
`vpi`	number	no	`8`	PPPoA VPI
`atmdev`	number	no	`0`	Specifies the ATM adapter number starting with 0. Most systems only have one ATM device and do not need this option
`encaps`	string	no	`llc`	PPPoA encapsulation mode: 'llc' (LLC) or 'vc' (VC)
`payload`	string	no	`bridged`	PPPoA forwarding mode: 'routed' or 'bridged'

DSL / VDSL

(!) This currently only works on devices based on lantiq SoCs.

(V)DSL uses a special config section called `dsl`, which typically looks like this:

```
config vdsl 'dsl'
        option annex 'b'
        option firmware '/lib/firmware/vdsl.bin'
        option tone 'bv'
        option xfer_mode 'atm'
```

The `dsl` section allows the following options:

Name	Type	Required	Default	Description
`annex`	string	yes	`b`	Specifies the Annex setting (ISP/line dependent). Supported values on lantiq AMAZON and DANUBE devices: b, bdmt, b2, b2p, a, at1, alite, admt, a2, a2p, l, m, m2, m2p Supported values on lantiq ARX100 "AR9" and VRX200 "VR9" devices: a, b, j
`firmware`	string	yes	`/lib/firmware/vdsl.bin`	The path to the modem's firmware image (!) **Only supported by devices with lantiq SoC.** See the xDSL firmware section below for more information
`tone`	string	yes	`bv`	The tone mode (ISP/line dependent). Supported values: a = A43, av = A43 + V43, b = B43, bv = B43 + V43 (!) **Only supported by devices with ARX100 "AR9" and VRX200 "VR9" lantiq SoC.** **This configuration was removed in "Designated Driver" as the driver now auto-detects the correct value**
`xfer_mode`	string	yes	`atm`	The transfer mode. Supported values are: atm = Asynchronous Transfer Mode (often used for ADSL connections), ptm = Packet Transfer Mode (often used for VDSL connections) (!) **Only supported by devices with ARX100 "AR9" and VRX200 "VR9" lantiq SoC.**

Lantiq xDSL firmware

Starting with r47631 and r47650 (lantiq: add dsl-vr9-firmware-xdsl / lantiq: add dsl-vrx200-firmware-xdsl-b: add Annex B version of VRX200 DSL firmware) there are redistributable versions of the xDSL firmware available as OpenWrt packages:

- dsl-vrx200-firmware-xdsl-a
- dsl-vrx200-firmware-xdsl-b

A list (incomplete) of other firmware versions, including those with vectoring support, can be found here: https://xdarklight.github.io/lantiq-xdsl-firmware-info/ [https://xdarklight.github.io/lantiq-xdsl-firmware-info/]

Aliases

Aliases: the old way

The "config alias" approach is *deprecated*. it used to be needed when multiple interfaces sharing the same device where not supported. JoW [https://forum.openwrt.org/viewtopic.php?pid=203943#p203943]

Alias sections can be used to define further IPv4 and IPv6 addresses for interfaces. They also allow combinations like DHCP on the main interface and a static IPv6 address in the alias, for example to deploy IPv6 on wan while keeping normal internet connectivity. Each interface can have multiple aliases attached to it.

A minimal alias declaration consists of the following lines:

```
config 'alias'
        option 'interface' 'lan'
        option 'proto' 'static'
        option 'ipaddr' '10.0.0.1'
        option 'netmask' '255.255.255.0'
```

- `lan` is the *logical interface name* of the parent interface
- `static` is the *alias interface protocol*
- `10.0.0.1` specifies the *alias ip address*
- `255.255.255.0` specifies the *alias netmask*

At the time of writing, only the `static` protocol type is allowed for aliases. Defined options for `alias` sections are listed below.

Name	Type	Required	Default	Description
`interface`	string	yes	*(none)*	Specifies the *logical interface name* of the parent (or master) interface this alias belongs to; must refer to one of the defined `interface` sections
`proto`	string	yes	*(none)*	Specifies the *alias interface protocol*
`ipaddr`	ip address	yes, if no `ip6addr` is set	*(none)*	IP address
`netmask`	netmask	yes, if no `ip6addr` is set	*(none)*	Netmask
`gateway`	ip address	no	*(none)*	Default gateway
`broadcast`	ip address	no	*(none)*	Broadcast address (autogenerated if not set)
`ip6addr`	ipv6 address	yes, if no `ipaddr` is set	*(none)*	IPv6 address (CIDR notation)
`ip6gw`	ipv6 address	no	*(none)*	IPv6 default gateway
`dns`	list of ip addresses	no	*(none)*	DNS server(s)
`layer`	integer	no	`3`	Selects the interface to attach to for stacked protocols (tun over bridge over eth, ppp over eth or similar). 3: attach to layer 3 interface (*tun**, *ppp** if parent is layer 3 else fallback to 2) 2: attach to layer 2 interface (*br-** if parent is bridge else fallback to layer 1) 1: attach to layer 1 interface (*eth**, *wlan**)

To list IP addresses associated with devices, you can run `ifconfig`, but that command will show only first IP address per device. However it will show *alias* device if you provided name (label) for it. The error proof method to list all IP addresses is:

```
ip addr
```

This "old" way works, at least, for OpenWrt 10.03.1 and 12.09.

Aliases: the new way

Basically create an 'interface' section per IP, but alias interfaces may NOT be of type bridge

- For non-bridged interfaces (physdev , that is physical interfaces) the `ifname` is the <interface-of-network-for-same-phydev>
- For cases where the interface is bridged the `ifname` is br-`base-interface`, where `base-interface` is the name of the primary IP's config section (e.g. for a the default lan interface config, the first alias would use ifname br-lan).

A minimal alias definition for a bridged interface might be (for a scenario without vlans):

```
config interface lan
        option 'ifname' 'eth0'
```

```
        option 'type' 'bridge'
        option 'proto' 'static'
        option 'ipaddr' '192.168.1.1'
        option 'netmask' '255.255.255.0'

config interface lan2
       option 'ifname' 'br-lan'
       option 'proto' 'static'
       option 'ipaddr' '10.0.0.1'
       option 'netmask' '255.255.255.0'
```

or for a non-bridge interface

```
config interface lan
        option 'ifname' 'eth0'
        option 'proto' 'static'
        option 'ipaddr' '192.168.1.1'
        option 'netmask' '255.255.255.0'

config interface lan2
       option 'ifname' 'eth0'
       option 'proto' 'static'
       option 'ipaddr' '10.0.0.1'
       option 'netmask' '255.255.255.0'
```

To see a list of interfaces you can do `ubus list network.interface.*` and to view the ip of a particular interface (the UCI name not the physical interface), do `ifstatus <interface>` (e.g. `ifstatus lan2`).

Does not work on OpenWRT 10.03.x .

Aliases: notes

On openwrt 12.09, a lan interface that is first defined as dhcp interface and then has aliases with static ip address could cause problems in routing the lan traffic through the wan zone using the basic lan-wan forwarding provided by openwrt. A solution is: having the basic interface with static address and aliases with dhcp protocol.

Another note is related to 'how to refer to the ifname of an interface'. Normally the ifname is `br-wan` if the interface `wan` is bridged, else is `ifname <nic_device>` . Another way to avoid to list always the same device is using `ifname @interface`. In this way, even if the wan interface is not a bridge, one can refer to the physical device used by the wan interface indirectly.

IPv4 Routes

Static *IPv4 routes* can be defined on specific interfaces using `route` sections. As for *aliases*, multiple sections can be attached to an interface.

A minimal example looks like this:

```
config 'route' 'name_your_route'
        option 'interface' 'lan'
        option 'target' '172.16.123.0'
        option 'netmask' '255.255.255.0'
        option 'gateway' '172.16.123.100'
```

- `lan` is the *logical interface name* of the parent interface
- `172.16.123.0` is the *network address* of the route
- `255.255.255.0` specifies the *route netmask*

Legal options for *IPv4 routes* are:

Name	Type	Required	Default	Description
`interface`	string	yes	*(none)*	Specifies the *logical interface name* of the parent (or master) interface this route belongs to; must refer to one of the defined `interface` sections
`target`	ip address	yes	*(none)*	Network address
`netmask`	netmask	no	*(none)*	Route netmask. If omitted, `255.255.255.255` is assumed which makes `target` a *host address*
`gateway`	ip address	no	*(none)*	Network gateway. If omitted, the `gateway` from the parent interface is taken; if set to `0.0.0.0` no gateway will be specified for the route
`metric`	number	no	`0`	Specifies the *route metric* to use
`mtu`	number	no	*interface MTU*	Defines a specific MTU for this route
`table`	routing table	no	*(none)*	Defines the table ID to use for the route. The ID can be either a numeric table index ranging from 0 to 65535 or a symbolic alias declared in /etc/iproute2/rt_tables. The special aliases local (255), main (254) and default (253) are recognized as well
`source`	ip address	no	*(none)*	The preferred source address when sending to destinations covered by the target
`onlink`	boolean	no	`0`	When enabled gateway is on link even if the gateway does not match any interface prefix (Barrier Breaker and later only)
`type`	string	no	`unicast`	One of the types outlined in the Routing Types table below (Barrier Breaker and later only)

To disable a route quickly, the option `enabled` is not available. Just rewrite the `route` config section as `disabled_route` like:

```
config 'disabled_route' 'name_your_route'
        ...lines...
```

and it will be recognized by the uci parser but not applied by the `/etc/init.d/network` script.

(!) It seems that on openwrt 12.09 if a route is defined using a gateway in an address space where a gateway is already defined, it will be not added. Like the lan has the gateway 192.168.1.1 and we want to go to 1.2.3.4 over the gateway 192.168.1.5 within the interface lan, it will not be added. Could be added through `ip route` commands tough.

IPv6 Routes

IPv6 routes can be specified as well by defining one or more `route6` sections.

A minimal example looks like this:

```
config 'route6'
        option 'interface' 'lan'
        option 'target' '2001:0DB8:100:F00:BA3::1/64'
        option 'gateway' '2001:0DB8:99::1'
```

- `lan` is the *logical interface name* of the parent interface
- `2001:0DB8:100:F00:BA3::1/64` is the routed *IPv6 subnet* in CIDR notation
- `2001:0DB8:99::1` specifies the *IPv6 gateway* for this route

Legal options for *IPv6 routes* are:

Name	Type	Required	Default	Description
`interface`	string	yes	*(none)*	Specifies the *logical interface name* of the parent (or master) interface this route belongs to; must refer to one of the defined `interface` sections
`target`	ipv6 address	yes	*(none)*	IPv6 network address
`gateway`	ipv6 address	no	*(none)*	IPv6 gateway. If omitted, the `gateway` from the parent interface is taken
`metric`	number	no	`0`	Specifies the *route metric* to use
`mtu`	number	no	*interface MTU*	Defines a specific MTU for this route
`table`	routing table	no	*(none)*	Defines the table ID to use for the route. The ID can be either a numeric table index ranging from 0 to 65535 or a symbolic alias declared in /etc/iproute2/rt_tables. The special aliases local (255), main (254) and default (253) are recognized as well
`source`	ip address	no	*(none)*	The preferred source address when sending to destinations covered by the target
`onlink`	boolean	no	`0`	When enabled gateway is on link even if the gateway does not match any interface prefix (Barrier Breaker and later only)
`type`	string	no	`unicast`	One of the types outlined in the Routing Types table below (Barrier Breaker and later only)

Routing Types

Type	Description
`unicast`	the route entry describes real paths to the destinations covered by the route prefix.
`local`	the destinations are assigned to this host. The packets are looped back and delivered locally.
`broadcast`	the destinations are broadcast addresses. The packets are sent as link broadcasts.
`multicast`	a special type used for multicast routing. It is not present in normal routing tables.
`unreachable`	these destinations are unreachable. Packets are discarded and the ICMP message host unreachable is generated. The local senders get an EHOSTUNREACH error.
`prohibit`	these destinations are unreachable. Packets are discarded and the ICMP message communication administratively prohibited is generated. The local senders get an EACCES error.
`blackhole`	these destinations are unreachable. Packets are discarded silently. The local senders get an EINVAL error.
`anycast`	the destinations are anycast addresses assigned to this host. They are mainly equivalent to local with one difference: such addresses are invalid when used as the source address of any packet.

IP rules

Since OpenWrt Barrier Breaker, netifd supports *IP rule* declarations which are required to implement policy routing.
IPv4 rules can be defined by declaring one or more sections of type `rule`, IPv6 rules are denoted by sections of type `rule6`. Both types share the same set of defined options.

A simple IPv4 rule may look like:

```
config rule
        option mark   '0xFF'
        option in     'lan'
        option dest   '172.16.0.0/16'
        option lookup '100'
```

- `0xFF` is a fwmark [http://www.tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.netfilter.html] to be matched
- `lan` is the incoming *logical interface name*
- `172.16.0.0/16` is the destination subnet to match
- `100` is the routing table ID to use for the matched traffic

Similary, an IPv6 rule looks like:

```
config rule6
        option in     'vpn'
        option dest   'fdca:1234::/64'
        option action 'prohibit'
```

- `vpn` is the incoming *logical interface name*
- `fdca:1234::/64` is the destination subnet to match
- `prohibit` is a routing action to take

The options below are defined for *IP rule* (`rule` and `rule6`) sections:

Name	Type	Required	Default	Description
`in`	string	no	*(none)*	Specifies the incoming *logical interface name*
`out`	string	no	*(none)*	Specifies the outgoing *logical interface name*
`src`	ip subnet	no	*(none)*	Specifies the source subnet to match (CIDR notation)
`dest`	ip subnet	no	*(none)*	Specifies the destination subnet to match (CIDR notation)
`tos`	integer	no	*(none)*	Specifies the TOS value to match in IP headers
`mark`	mark/mask	no	*(none)*	Specifies the *fwmark* and optionally its mask to match, e.g. `0xFF` to match mark 255 or `0x0/0x1` to match any even mark value
`invert`	boolean	no	`0`	If set to `1`, the meaning of the match options is inverted
`priority`	integer	no	*(incrementing)*	Controls the order of the IP rules, by default the priority is auto-assigned so that they are processed in the same order they're declared in the config file
`lookup`	routing table	at least one of	*(none)*	The rule target is a table lookup, the ID can be either a numeric table index ranging from `0` to `65535` or a symbolic alias declared in `/etc/iproute2/rt_tables`. The special aliases `local` (`255`), `main` (`254`) and `default` (`253`) are recognized as well
`goto`	rule index			The rule target is a jump to another rule specified by its `priority` value
`action`	string			The rule target is one of the routing actions outlined in the table below

Routing Actions

Action	Description
`prohibit`	When reaching the rule, respond with *ICMP prohibited* messages and abort route lookup
`unreachable`	When reaching the rule, respond with *ICMP unreachable* messages and abort route lookup
`blackhole`	When reaching the rule, drop packet and abort route lookup
`throw`	Stop lookup in the current routing table even if a default route exists

Examples

Below are a few examples for special, non-standard interface configurations.

Bridge without IP

```
config 'interface' 'example'
        option 'type'    'bridge'
        option 'proto'   'none'
        option 'ifname'  'eth0 eth1'
        option 'auto'    '1'
```

DHCP without default gateway

```
config 'interface' 'example'
        option 'proto'   'dhcp'
        option 'ifname'  'eth0'
        option 'defaultroute' '0'
```

Older versions of OpenWRT used this instead, but this method is deprecated and no longer works in Chaos Calmer:

```
config 'interface' 'example'
        option 'proto'   'dhcp'
        option 'ifname'  'eth0'
        option 'gateway' '0.0.0.0'
```

DHCP and IPv6

```
config 'interface' 'example'
        option 'proto'     'dhcp'
        option 'ifname'    'eth0'

config 'alias'
        option 'interface' 'example'
        option 'proto'     'static'
        option 'ip6addr'   '2001:0DB8:100:F00:BA3::1'
```

Static IP configuration with multiple dnses

```
config 'interface' 'example'
        option 'proto'     'static'
        option 'ifname'    'eth0'
        option 'ipaddr'    '192.168.1.200'
        option 'netmask'   '255.255.255.0'
        list   'dns'       '192.168.1.1'
        list   'dns'       '192.168.10.1'
        # the priority is: the last dns listed will be the first one
        # to be chosen for the name resolution.
```

(!) Openwrt will use the new dns configured only after a reboot or a `/etc/init.d/dnsmasq restart`.

Static IP configuration and default gateway with non-zero metric

```
config 'interface' 'example'
        option 'proto'     'static'
        option 'ifname'    'eth0'
        option 'ipaddr'    '192.168.1.200'
        option 'netmask'   '255.255.255.0'
        option 'dns'       '192.168.1.1'

config 'route'
        option 'interface' 'example'
        option 'target'    '0.0.0.0'
        option 'netmask'   '0.0.0.0'
        option 'gateway'   '192.168.1.1'
        option 'metric'    '100'
```

PPtP-over-PPPoE internet connection

```
config 'interface' 'wan'
        option 'proto'     'pppoe'
        option 'ifname'    'eth1'
        option 'username'  'user'
        option 'password'  'pass'
        option 'timeout'   '10'

config 'interface' 'vpn'
        option 'proto'     'pptp'
        option 'ifname'    'vpn'
        option 'username'  'vpnuser'
        option 'password'  'vpnpass'
        option 'server'    'vpn.example.org'
```

(!) Additionally the "wan" firewall zone must include both interfaces in `/etc/config/firewall`:

```
config 'zone'
        option 'name'      'wan'
        option 'network'   'wan vpn'  # Important
        option 'input'     'REJECT'
        option 'forward'   'REJECT'
        option 'output'    'ACCEPT'
        option 'masq'      '1'
```

PPPoA ADSL internet connection

```
config adsl-device 'adsl'
        option fwannex 'a'
        option annex 'a'

config interface 'wan'
        option proto 'pppoa'
        option username 'jbloggs@plusdsl.net'
        option password 'XXXXXXXXX'
        option vpi '0'
        option vci '38'
        option encaps 'vc'
```

listing an interface created by software on the router, like vpn

For example, a vpn interface is normally "tun0". To list it in the uci config files (and therefore in luci):

```
config interface 'tun0'
        option ifname 'tun0'
        option proto 'none'
```

Static IPv6-in-IPv4 tunnel

The example below illustrates a static tunnel configuration in `/etc/config/network` file for the Hurricane Electric (he.net) broker. Option `ipaddr` specifies the local IPv4 address, `peeraddr` is the broker IPv4 address and `ip6addr` the local IPv6 address routed via the tunnel.

```
config 'interface' 'henet'
        option 'proto'     '6in4'
        option 'ipaddr'    '178.24.115.19'
        option 'peeraddr'  '216.66.80.30'
        option 'ip6addr'   '2001:0DB8:1f0a:1359::2/64'
```

(!) You should also add an address from your routed IPv6 network to the "lan" interface.

(!) To apply IPv6 firewall rules to the tunnel interface, add it to the "wan" zone in `/etc/config/firewall`:

```
config 'zone'
        option 'name'      'wan'
        option 'network'   'wan henet'  # Important
        option 'input'     'REJECT'
        option 'forward'   'REJECT'
        option 'output'    'ACCEPT'
        option 'masq'      '1'
```

(!) If you define a new, dedicated zone just for the tunnel interface, make sure to set `option conntrack 1` in order to force enabling connection tracking, otherwise unidirectional forwarding rules will not work.

(!) Don't forget to set up forwarding rules between the LAN and the tunnel if you want to route IPv6 traffic between them.

Setup behind one-to-one NAT

If your public IP [http://checkip.dyndns.org/], e.g. `178.24.115.19`, is not matching the IP address on your WAN interface, your ISP is probably using one-to-one NAT [http://shorewall.net/NAT.htm#One-to-one] (aka full-cone NAT [http://en.wikipedia.org/wiki/Network_address_translation#Methods_of_Port_translation]) and you won't be able to establish static IPv6-in-IPv4 tunnel. IP address of your WAN interface can be obtained with the following command:

- Backfire

  ```
  uci -P/var/state get network.wan.ipaddr
  ```

- Trunk/Attitude Adjustment

  ```
  . /lib/functions/network.sh; network_get_ipaddr ip wan; echo $ip
  ```

If this is your case you should fill the WAN IP address into `ipaddr` option instead of your actual public IP that might have been provided to Hurricane Electric [http://he.net/] during tunnel creation.[1] Or you may completely omit the optional `ipaddr` option and let auto configuration to handle the correct IP. (Fix Me! Auto configuration is vague. Is `uci` handling this case?) That would be preferred solution if your WAN IP is dynamic (i.e. obtained via DHCP) or you are not sure. Example of `/etc/config/network` entry:

```
config 'interface' 'henet'
        option 'proto'     '6in4'
        option 'peeraddr'  '216.66.80.30'
        option 'ip6addr'   '2001:0DB8:1f0a:1359::2/64'
```

Note: you could probably try to define alias for WAN interface with your public IP address. Then you could use your public IP in `ipaddr` option and system would find its way to your WAN interface that has only private IP address because of the one-to-one NAT. (Fix Me! However, it didn't really worked for me. I got this advice on IRC and it looks reasonable, thats why I put it here anyway. If it was **not** supposed to fix it, just delete this note.)

Dynamic IPv6-in-IPv4 tunnel (HE.net only)

The example below illustrates a dynamic tunnel configuration for the Hurricane Electric (he.net) broker with enabled IP update. The local IPv4 address is automatically determined and tunnelid, username and password are provided for IP update.

```
config 'interface' 'henet'
        option 'proto'     '6in4'
        option 'peeraddr'  '216.66.80.30'
        option 'ip6addr'   '2001:0DB8:1f0a:1359::2/64'
        option 'tunnelid'  '12345'
        option 'username'  'myusername'
        option 'password'  '098f6bcd4621d373cade4e832627b4f6'
```

(!) You should also add an address from your routed IPv6 network to the "lan" interface.

(!) To apply IPv6 firewall rules to the tunnel interface, add it to the "wan" firewall zone, see example above for details.

(!) The password entered above should be the md5sum of the password you use to log in to tunnelbroker.net.

L2TPv3 Pseudowire bridged to LAN

This example establishes a Pseudowire Tunnel and bridges it to the LAN ports. The existing lan interface is reused with protocol `l2tp` instead of `static`.

```
config 'interface' 'lan'
        option 'proto'     'l2tp'
        option 'type'      'bridge'
        option 'ifname'    'eth0'
        option 'ipaddr'    '192.168.1.1'
        option 'netmask'   '255.255.255.0'
        option 'localaddr' '178.24.154.19'
        option 'peeraddr'  '89.44.33.61'
        option 'encap'     'udp'
        option 'sport'     '4000'
        option 'dport'     '5410'
```

Relay between LAN and Wireless Station

This example sets up a `relayd` pseudo bridge between a wireless client network and LAN, so that it works similarly to the Broadcom Bridged Client mode.

Wireless configuration (excerpt):

```
config wifi-iface
        option 'device'     'radio0'
        option 'mode'       'sta'
        option 'ssid'       'Some Wireless Network'
        option 'encryption' 'psk2'
        option 'key'        '12345678'
        option 'network'    'wwan'
```

Network configuration (excerpt):
(!) Note that the LAN subnet must be different from the one used by wireless network's DHCP.

```
config 'interface' 'lan'
        option 'ifname'     'eth0.1'
        option 'proto'      'static'
        option 'ipaddr'     '192.168.1.1'
        option 'netmask'    '255.255.255.0'

config 'interface' 'wwan'
        option 'proto'      'dhcp'

config 'interface' 'stabridge'
        option 'proto'      'relay'
        option 'network'    'lan wwan'
```

In contrast to true bridging, traffic forwarded in this manner is affected by firewall rules, therefore both the wireless client network and the lan network should be covered by the same LAN firewall zone with forward policy set to `accept` to allow traffic flow between both interfaces:

```
config 'zone'
        option 'name'        'lan'
        option 'network'     'lan wwan'  # Important
        option 'input'       'ACCEPT'
        option 'forward'     'ACCEPT'    # Important
        option 'output'      'ACCEPT'
```

Static addressing of a GRE tunnel

Create a GRE tunnel with static address 10.42.0.253/30, adding it to an existing firewall zone called `tunnels`:

```
config interface mytunnel
        option proto    gre
        option zone     tunnels
        option peeraddr 198.51.100.42

config interface mytunnel_addr
        option proto    static
        option ifname   @mytunnel
        option ipaddr   10.42.0.253
        option netmask  255.255.255.252
        # Fixes IPv6 multicast (long-standing bug in kernel).
        # Useful if you run Babel or OSPFv3.
        option ip6addr  'fe80::42/64'
```

Network management

The complete network configuration can be re-applied by running `/etc/init.d/network restart`. Individual interfaces can be brought up with `ifup` *`name`* or down with `ifdown` *`name`* where *name* corresponds to the *logical interface name* of the corresponding `config interface` section. An `ifup` implies a prior `ifdown` so there is no need to invoke both when reloading an interface.

Note that wireless interfaces are managed externally and `ifup` may break the relation to existing bridges. In such a case it is required to run `wifi up` after `ifup` in order to re-establish the bridge connection.

Determining Linux interface names

In order to derive a Linux interface name like `eth1` from a logical network name like `wan` for use in scripts or tools like `ifconfig` and `route` the `uci` utility can be used as illustrated in the example below which opens port 22 on the interface.

```
WANIF=$(uci -P/var/state get network.wan.ifname)
iptables -I INPUT -i $WANIF -p tcp --dport 22 -j ACCEPT
```

The uci state vars are deprecated and not used anymore for network related information Quoting jow in the forum [https://forum.openwrt.org/viewtopic.php?pid=203787#p203787]. Use /lib/functions/network.sh:

```
source /lib/functions/network.sh

if network_get_ipaddr addr "wan"; then
    echo "IP is $addr"
fi
```

Multiple IP addresses

Assigning multiple ip addresses to the same interface:

```
config interface foo
  option ifname eth1
  list ipaddr 10.8.0.1/24
  list ipaddr 10.9.0.1/24
  list ip6addr fdca:abcd::1/64
  list ip6addr fdca:cdef::1/64
```

Specifying multiple interfaces sharing the same device:

```
config interface foo
  option ifname eth1
  option ipaddr 10.8.0.1
  option netmask 255.255.255.0
  option ip6addr fdca:abcd::1/64

config interface foo2
  option ifname eth1
  option ipaddr 10.9.0.1
  option netmask 255.255.255.0
  option ip6addr fdca:cdef::1/64
```

More info at https://dev.openwrt.org/ticket/2829#comment:7 [https://dev.openwrt.org/ticket/2829#comment:7].

1)
You should always use your public IP while creating Hurricane Electric tunnel, so don't change it just because you are behind one-to-one NAT.