

# **Package IPV6 - IPv6-Enhancement for fli4l Routers Version 3.10.5**

Christoph Schulz                      the fli4l-Team  
email: [fli4l@kristov.de](mailto:fli4l@kristov.de)              email: [team@fli4l.de](mailto:team@fli4l.de)

February 16, 2016

# Contents

<b>1. Documentation Of Package IPV6</b>	<b>3</b>
1.1. IPv6 - Internet Protocol Version 6 . . . . .	3
1.1.1. Introduction . . . . .	3
1.1.2. Address Format . . . . .	3
1.1.3. Configuration . . . . .	4
1.1.4. Web-GUI . . . . .	15
<b>A. Appendix Of Package IPV6</b>	<b>16</b>
A.1. IPV6 - Connection to IPv6-Internet using a SixXS-Tunnel . . . . .	16
A.1.1. Get An Account . . . . .	16
A.1.2. Tunnel Configuration . . . . .	16
A.1.3. Configuration Of The Subnet . . . . .	18
<b>List of Figures</b>	<b>22</b>
<b>List of Tables</b>	<b>23</b>
<b>Index</b>	<b>24</b>

# 1. Documentation Of Package IPV6

## 1.1. IPv6 - Internet Protocol Version 6

### 1.1.1. Introduction

This package enables fli4l to support IPv6 in many ways. This includes informations about the IPv6 address of the router's IPv6 (sub) networks managed by it, predefined IPv6 routes and firewall rules regarding IPv6 packets. Further other IPv6-based services can be provided, such as configuration via DHCPv6. Last but no least it is possible to automatically establish tunnels to IPv6 providers. This currently works only with so-called 6in4 tunneling as supported by the provider "SixXS". Other technologies (AYIYA, 6to4, Teredo) are not supported at the moment.

IPv6 is the successor of the internet protocol IPv4. It was developed mainly to enlarge the relatively small amount of unique internet addresses: While IPv4 supports approximately  $2^{32}$  addresses, <sup>1</sup> in IPv6  $2^{128}$  are possible. Each communicating IPv6 host can thus have a unique address and no longer has to rely on techniques such as NAT, PAT, masquerading a.s.o.

Besides this aspect topics such as self-configuration and safety concerns also played a role in the development of the IPv6 protocol. This will be carried out in later sections.

The biggest problem with IPv6 is its spread: Currently IPv6 –compared with IPv4 – is rarely used. The reason is that IPv6 and IPv4 are technically incompatible and thus all software and hardware components involved in packet forwarding in the internet have to be retrofitted for IPv6. Certain services such as DNS (Domain Name System) must be extended in accordance with IPv6.

This is a vicious circle: The low spreading of IPv6 with service providers on the internet leads to disinterest on the part of router manufacturers to equip their devices with IPv6 functionality, which in turn means that service providers fear the transition to IPv6 because they fear that it's not worth the effort. Only slowly the mood is changing in favour of IPv6 mostly because of the increasing pressure from shortness of IPv4 address supplies. <sup>2</sup>

### 1.1.2. Address Format

An IPv6 address consists of eight two-byte values listed in hexadecimal notation:

*Example 1:* 2001:db8:900:551:0:0:0:2

*Example 2:* 0:0:0:0:0:0:0:1 (IPv6-loopback-address)

To make the addresses a little clearer, successive zeros are merged by removing them, and only two successive colons remain. The above addresses may therefore be rewritten as:

*Example 1 (compact):* 2001:db8:900:551::2

*Example 2 (compact):* ::1

---

<sup>1</sup>only nearly because some addresses are used for specialized purposes, such as for broadcast and multicasting

<sup>2</sup>The last IPv4 address blocks have been assigned by the IANA by now.

Such a reduction is only allowed once to avoid ambiguities. The address 2001:0:0:1:2:0:0:3 can thus either be shortened to 2001::1:2:0:0:3 or 2001:0:0:1:2::3 but not to 2001::1:2::3 because then it would be unclear how the four zeros belong to the contracted regions.

Another ambiguity exists when an IPv6 address is to be combined with a port (TCP or UDP): In this case you can not add the port immediately followed by a colon and value because the colon is already used within the address and it is therefore unclear in some cases whether the port specification is perhaps an address component. In such cases the IPv6 address is enclosed in square brackets. This is the syntax also required in URLs (for example if a numerical IPv6 address should be used in the web browser).

*Example 3:* [2001:db8:900:551::2]:1234

Without the use of braces the following address is created 2001:db8:900:551::2:1234, which without shortening equals to the address 2001:db8:900:551:0:0:2:1234 and thus has no port appendend.

### 1.1.3. Configuration

#### General Settings

The general settings include at first the activation of IPv6 support and then the optional assignment of an IPv6 address to the router.

**OPT\_IPV6** Activates IPv6 support.

Default setting: OPT\_IPV6='no'

**HOSTNAME\_IP6** (optional) This variable sets the IPv6-address of the routers explicitly. If the variable is not set the IPv6-address will be set to the first configured IPv6-subnet (IPV6\_NET\_x, see below).

Example: HOSTNAME\_IP6='IPV6\_NET\_1\_IPADDR'

#### Subnet Configuration

This section describes the configuration of one or more IPv6 subnets. An IPv6 subnet is an IPv6 address space specified by a so-called prefix and is bound to a certain network interface. Further settings include the announcing of the prefix and of the DNS service within the subnet and optional a router name within that subnet.

**IPV6\_NET\_N** This variable contains the number of used IPv6 subnets. At least one IPv6 subnet should be defined in order to use IPv6 in the local network.

Default setting: IPV6\_NET\_N='0'

**IPV6\_NET\_x** This variable contains the IPv6 address of the router and the size of the network mask in CIDR notation for a specific IPv6 subnet. If the subnet is to be routed publicly it usually is obtained from the Internet or tunnel provider.

**Important:** *If the stateless auto-configuration should be activated in the subnet then the length of the subnet prefix has to be 64 bits! (see the section on IPV6\_NET\_x\_ADVERTISE below)*

## 1. Documentation Of Package IPV6

**Important:** *If the subnet is connected to a tunnel (see `IPV6_NET_x_TUNNEL` below) then only the part of the router address is specified here that is not assigned to the tunnel's subnet prefix (to be found in `IPV6_TUNNEL_x_PREFIX`) because that prefix and the address are combined! In previous versions of the IPv6 package the variable `IPV6_TUNNEL_x_PREFIX` did not exist and subnet prefix and router address were combined together in `IPV6_NET_x`. That does not work if the subnet prefix associated by the tunnel provider is first handed out when instantiating the dynamic tunnel. In addition the length of the subnet prefix (in this case /48) is not known and certain predefined routes can not be set properly. This leads to strange effects while routing to some specific destinations. The configuration was separated to avoid such effects.*

Examples:

```
IPV6_NET_1='2001:db8:1743:42::1/64'      # without tunnel: complete address
IPV6_NET_1_TUNNEL=''

IPV6_NET_2='0:0:0:42::1/64'              # with tunnel: partial address
IPV6_NET_2_TUNNEL='1'
IPV6_TUNNEL_1_PREFIX='2001:db8:1743::/48' # see "Tunnel Configuration"
```

**IPV6\_NET\_x\_DEV** This variable contains for a specific IPv6 subnet the name of the network interface to which the IPv6 network is bound. This does *not* collide with the network interfaces assigned in the base configuration (`base.txt`) because a network interface can have both IPv4 and IPv6 addresses assigned.

Example: `IPV6_NET_1_DEV='eth0'`

**IPV6\_NET\_x\_TUNNEL** This variable contains the index of the associated tunnel for a specific IPv6 subnet. The prefix of the tunnel is combined with the router address to get the complete IPv6 address of the router. If the variable is empty or undefined then no tunnel belongs to the subnet and in `IPV6_NET_x` the full IPv6 address of the router including network mask has to be specified (see above).

A tunnel can be assigned to multiple subnets because a tunnel subnet is usually large enough to be split in multiple subnets (/56 or greater). It is not possible to assign multiple tunnel subnet prefixes to a subnet because the address of the subnet would be ambiguous in this case.

Example: `IPV6_NET_1_TUNNEL='1'`

**IPV6\_NET\_x\_ADVERTISE** This variable determines whether the chosen subnet prefix is advertised on the LAN via “Router Advertisements”. This is used for the so-called “Stateless auto-configuration” and should not be confused with DHCPv6. Possible values are “yes” and “no”.

It is recommended to enable this setting, unless all addresses in the network are statically assigned or another router is already responsible to advertise the subnet prefix.

**Important:** *Automatic distribution of the subnet will only work if the subnet is a /64 network, i.e. if the length of the subnet prefix is 64 bits! The reason for this is that the other hosts in the network compute their IPv6 address from the prefix and their host MAC*

## 1. Documentation Of Package IPV6

*addresses which will not work if the host part is not 64 bits. If the self-configuration fails the subnet prefix should be checked for incorrect length (for example as /48).*

Default setting: `IPV6_NET_1_ADVERTISE='yes'`

**IPV6\_NET\_x\_ADVERTISE\_DNS** This variable determines whether the local DNS service in IPv6 subnets should be advertised by “Router Advertisements”. This will only work if the IPv6 functionality of the DNS service is activated using `DNS_SUPPORT_IPV6='yes'`. Possible values are “yes” and “no”.

Default setting: `IPV6_NET_1_ADVERTISE_DNS='no'`

**IPV6\_NET\_x\_DHCP** This variable enables the DHCPv6 service for this IPv6 subnet. Possible values are “yes” and “no”. DHCPv6 will only be used to provide information on domain names and address of the used DNS server to hosts in this subnet. The assignment of IPv6 addresses via DHCPv6 is currently not possible with fli4l.

The address of the DNS server is published via DHCPv6 only if the IPv6 support of the DNS service is enabled via `DNS_SUPPORT_IPV6` in package `dns_dhcp`.

**Important:** *The variables `IPV6_NET_x_ADVERTISE_DNS` and `IPV6_NET_x_DHCP` do not exclude each other and can both be enabled. In this case the address of the DNS server can be queried in two ways by hosts on the local network.*

**Per network interface a maximum of one bound IPv6-subnet can be configured for DHCPv6!**

Default setting: `IPV6_NET_1_DHCP='no'`

**IPV6\_NET\_x\_NAME** (optional) This variable specifies an interface-specific hostname for the router in this IPv6-subnet.

Example: `IPV6_NET_1_NAME='fli4l-subnet1'`

### Tunnel Configuration

This section introduces the configuration of 6in4 IPv6 tunnels. Such a tunnel is useful in case that your own ISP does not support IPv6 natively. With an internet host or a tunnel broker known as a PoP (Point of Presence) a bi-directional link via IPv4 can be established. All IPv6 packets will be routed encapsulated (therefore 6 “in” 4 because of the IPv6 packets are encapsulated in IPv4 packets).<sup>3</sup> First the tunnel will be established and in a second step the router is configured in a way that the IPv6 packets to the Internet are routed through the tunnel. The first part is covered in this section and the second part is described in the next section.

**IPV6\_TUNNEL\_N** This variable contains the number of 6in4 tunnels to be established.

Example: `IPV6_TUNNEL_N='1'`

---

<sup>3</sup>This is known as IPv4 protocol 41, “IPv6 encapsulation”.

**IPV6\_TUNNEL\_x\_TYPE** This variable determines the type of the tunnel. Currently, the values “raw”, “static”, “sixxs” for dynamic heartbeat-tunnels by the provider SixXS and “he” for tunnels of provider Hurricane Electric are supported. More about heartbeat-tunnels in the next paragraph.

Example: `IPV6_TUNNEL_1_TYPE='sixxs'`

**IPV6\_TUNNEL\_x\_DEFAULT** This variable determines whether IPv6 packets which are not addressed to local networks are allowed to be routed through this tunnel. Only one tunnel can do this (because there is only one default route). Possible values are “yes” and “no”.

**Important:** *Exactly one tunnel should be a default gateway for outbound IPv6 otherwise no communication with IPv6 hosts on the Internet is possible! The use of only non-default tunnels is only useful if outgoing IPv6 traffic is sent on a separately configured default route, which is not related to a tunnel. See the introduction to the subsection “Route Configuration” and the description of the variable `IPV6_ROUTE_x` below.*

Default setting: `IPV6_TUNNEL_1_DEFAULT='no'`

**IPV6\_TUNNEL\_x\_PREFIX** This variable contains the IPv6-subnet-prefix of the tunnel in CIDR-Notation which means an IPv6-address is provided as well as the length of the prefix. This settings are usually given to you by the tunnel provider. This setting is not necessary for tunnel providers giving a new prefix each time a tunnel is established. (Such providers are not supported at the moment.) This variable has to be empty with “raw” tunnels as well.

**Important:** *This variable may be empty if the tunnel has no subnet-prefix assigned. But in turn the tunnel can not be assigned to a IPv6-subnet (`IPV6_NET_x`) because IPv6-addresses in the subnet ca not be computed. Such a configuration makes only sense on an interim basis if the tunnel has to be active for some time before the tunnel provider assigns a subnet-prefix (this is the usual procedure for the tunnel provider SixXS).*

Examples:

```
IPV6_TUNNEL_1_PREFIX='2001:db8:1743::/48'      # /48-subnet
IPV6_TUNNEL_2_PREFIX='2001:db8:1743:5e00::/56'  # /56-subnet
```

**IPV6\_TUNNEL\_x\_LOCALV4** This variable contains the local IPv4-address of the tunnel or the value ‘dynamic’ if the dynamic IPv4-address of the active WAN-Circuits should be used. The latter makes only sense for a heartbeat-tunnel (see `IPV6_TUNNEL_x_TYPE` below).

Examples:

```
IPV6_TUNNEL_1_LOCALV4='172.16.0.2'
IPV6_TUNNEL_2_LOCALV4='dynamic'
```

**IPV6\_TUNNEL\_x\_REMOTEV4** This variable contains the remote IPv4-address of the tunnel. Usually this value is given to you by the tunnel provider.

Example (as used by PoP deham01 by Easynet):

```
IPV6_TUNNEL_1_REMOTEV4='212.224.0.188'
```

**Important:** *If PF\_INPUT\_ACCEPT\_DEF is set to “no” (the IPv4-firewall is configured manually) you will need a firewall rule to accept all IPv6-in-IPv4 packets (IP-Protokoll 41) from the tunnel endpoint. For the tunnel endpoint above the rule would be like this:*

```
PF_INPUT_x='prot:41 212.224.0.188 ACCEPT'
```

**IPV6\_TUNNEL\_x\_LOCALV6** This variable sets the local IPv6 address of the tunnel including the used netmask in CIDR notation. This information is predetermined from the tunnel provider. This information is unnecessary for tunnel providers which re-assign the tunnel endpoints for establishing the tunnel new each time (Such Providers are not yet supported).

Example: `IPV6_TUNNEL_1_LOCALV6='2001:db8:1743::2/112'`

**IPV6\_TUNNEL\_x\_REMOTEV6** This variable specifies the remote IPv6 address of the tunnel. This information is set by the tunnel provider. A netmask is not needed because it is taken from the variable `IPV6_TUNNEL_x_LOCALV6`. This information is unnecessary for tunnel providers assigning new tunnel endpoints for every tunnel established (Such providers are not yet supported).

Example: `IPV6_TUNNEL_1_REMOTEV6='2001:db8:1743::1'`

**IPV6\_TUNNEL\_x\_DEV** (optional) This variable contains the name of the network interface of the tunnel to be created. Different tunnels have to be named differently to make everything work. If the variable is not defined a tunnel name is generated automatically (“v6tun” + tunnel index).

Example: `IPV6_TUNNEL_1_DEV='6in4'`

**IPV6\_TUNNEL\_x\_MTU** (optional) This variable contains the size of the MTU (Maximum Transfer Unit) in bytes, i.e. the size of the largest packet that can still be tunneled. This information is generally predetermined by the tunnel provider. If nothing is specified the standard setting is “1280” and should work with all tunnels.

Default setting: `IPV6_TUNNEL_1_MTU='1280'`

To prevent a host blocking a tunnel although it does not need it at all some providers require permanently sending packets over the tunnel to the provider to prove it is still “alive”. For this purpose a so-called heartbeat protocol is used. Providers usually require a successful login with user name and password in order to avoid abuse. If such a heartbeat tunnel is used (as offered by SixXS) then appropriate information has to be passed which will be described below.

**IPV6\_TUNNEL\_x\_USERID** This variable holds the username needed for the tunnel login.

Example: `IPV6_TUNNEL_1_USERID='ABCDE-SIXXS'`



**IPV6\_TUNNEL\_x\_PASSWORD** This variable contains the password for the username above. It can't contain spaces.

Example: `IPV6_TUNNEL_1_PASSWORD='password'`

**IPV6\_TUNNEL\_x\_TUNNELID** This variable contains the identifier for the tunnel. The name of a SixXS tunnel always starts with a capital 'T'.

Example: `IPV6_TUNNEL_1_TUNNELID='T1234'`

**IPV6\_TUNNEL\_x\_TIMEOUT** (optional) This variable contains the maximum waiting time in seconds for the tunnel to establish. The default setting depends on the tunnel provider.

Example: `IPV6_TUNNEL_1_TIMEOUT='30'`

## Route Configuration

Routes are paths for IPv6 packets. To know the direction in which it should send an incoming packet the router does rely on a routing table in which this information can be found. In the case of IPv6 it is important to know where IPv6 packets have to be sent that are not bound to the local network. A default route is automatically configured that sends all packets to the other end of an IPv6 tunnel if `IPV6_TUNNEL_x_DEFAULT` is set for the relevant tunnel. Other routes can be configured here (i.e. to interconnect parallel IPv6 subnets).

**IPV6\_ROUTE\_N** This variable sets the number of IPv6 routes to define. Usually additional IPv6 routes are not needed.

Default setting: `IPV6_ROUTE_N='0'`

**IPV6\_ROUTE\_x** This variable holds the route in form of 'target-net gateway'. The target net has to be specified in CIDR-notation. For the default route the target net has to be `::/0`. It is not necessary to configure default routes here that cross a tunnel (see introduction on this paragraph).

Example: `IPV6_ROUTE_1='2001:db8:1743:44::/64_2001:db8:1743:44::1'`

## IPv6 Firewall

As for IPv4 a firewall is needed for IPv6 networks in order to avoid that everyone can reach each machine in the local network from outside. This is even more important because every computer in IPv6 normally has a globally unique address which can be permanently assigned to the machine since it is computed from the MAC address of the network card.<sup>4</sup> Therefore the firewall blocks all requests from outside at first and can then be opened by corresponding entries in this section bit by bit as needed.

The configuration of an IPv6 firewall corresponds widely to that of an IPv4 firewall. Special features and differences will be explained separately.

**PF6\_LOG\_LEVEL** For all chains following the protocol setting in `PF6_LOG_LEVEL` is active. Possible values are: debug, info, notice, warning, err, crit, alert, emerg.

---

<sup>4</sup>An exception exists if the LAN hosts have activated so-called "Privacy Extensions" because then a part of the IPv6 address is generated randomly. These addresses are by definition not known outside and thus only partially or not at all relevant for the firewall configuration.

**PF6\_INPUT\_POLICY** This variable sets the default strategy for all incoming packets for the router (INPUT-Chain). Possible values are “REJECT” (default: rejects all packets), “DROP” (discards all packets without further notice) and “ACCEPT” (accepts all packets). For a detailed description see the documentation of PF\_INPUT\_POLICY.

Default setting: PF6\_INPUT\_POLICY='REJECT'

**PF6\_INPUT\_ACCEPT\_DEF** This variable activates the predefined rules for the INPUT-chain of the IPv6-Firewall. Possible values are “yes” and “no”.

The default rules open the firewall for incoming ICMPv6 pings (one ping per second as a limit) as well as for NDP packets (Neighbor Discovery Protocol) needed for stateless auto-configuration of IPv6 networks. Connections from localhost and response packets to locally initiated connections are also allowed. Finally the IPv4 firewall is adapted so that for each tunnel IPv6-in-IPv4 encapsulated packets are accepted by the end of the tunnel.

Default setting: PF6\_INPUT\_ACCEPT\_DEF='yes'

**PF6\_INPUT\_LOG** This variable activates logging of all rejected incoming packets. Possible values are “yes” and “no”. For a detailed description see documentation of PF\_INPUT\_LOG.

Default setting: PF6\_INPUT\_LOG='no'

**PF6\_INPUT\_LOG\_LIMIT** This variable configures the log-limit of the INPUT-chains of the IPv6-firewall to keep logfiles readable. For a detailed description see documentation of PF\_INPUT\_LOG\_LIMIT.

Default setting: PF6\_INPUT\_LOG\_LIMIT='3/minute:5'

**PF6\_INPUT\_REJ\_LIMIT** This variable sets the limit for rejection of incoming TCP-packets. If such a packet exceeds this limit the packet will be dropped silently (DROP). For a detailed description see the documentation of PF\_INPUT\_REJ\_LIMIT.

Default setting: PF6\_INPUT\_REJ\_LIMIT='1/second:5'

**PF6\_INPUT\_UDP\_REJ\_LIMIT** This variable sets the limit for rejection of incoming UDP-packets. If such a packet exceeds this limit the packet will be dropped silently (DROP). For a detailed description see the documentation of PF\_INPUT\_UDP\_REJ\_LIMIT.

Default setting: PF6\_INPUT\_UDP\_REJ\_LIMIT='1/second:5'

**PF6\_INPUT\_ICMP\_ECHO\_REQ\_LIMIT** Defines how often the router should react to ICMPv6-echo-queries. The frequency is written as ‘n/time period’ with bursts i.E. ‘3/minute:5’ (in analogy to the limit-restriction). The packet will be ignored (DROP) if the limit is reached. If empty the default setting ‘1/second:5’ will be used, if set to ‘none’ no limitations are accomplished.

Default setting: PF6\_INPUT\_ICMP\_ECHO\_REQ\_LIMIT='1/second:5'

**PF6\_INPUT\_ICMP\_ECHO\_REQ\_SIZE** Defines the maximum size of a received ICMPv6-echo-request (in Bytes). Beside the data the packet-header size has to be considered. The default value is 150 Bytes.

Default setting: PF6\_INPUT\_ICMP\_ECHO\_REQ\_SIZE='150'

**PF6\_INPUT\_N** This variable contains the number of IPv6-firewall rules for incoming packets (INPUT-Chain). Per default two rules are activated: the first allows all local hosts to access the router on so-called link-level addresses, the second allows hosts from the first defined IPv6-subnet to access the router.

In case of multiple local IPv6-subnets defined the second rule has to be cloned respectively. See the configuration file for details.

Example: `PF6_INPUT_N='2'`

**PF6\_INPUT\_x** This variable specifies a rule for the INPUT-chain of the der IPv6-firewall. For a detailed description see the documentation of `PF_INPUT_x`.

Differences regarding the IPv4-firewall:

- `IPV6_NET_x` has to be used instead of `IP_NET_x`.
- `IPV6_ROUTE_x` has to be used instead of `IP_ROUTE_x`.
- IPv6-addresses must be enclosed in square brackets (including the network mask, if present).
- All IPv6 address strings (including `IP_NET_x` etc.) must be enclosed in square brackets if a port or a port range follows.

Examples:

```
PF6_INPUT_1='[fe80::0/10] ACCEPT'
PF6_INPUT_2='IPV6_NET_1 ACCEPT'
PF6_INPUT_3='tmp1:samba DROP NOLOG'
```

**PF6\_INPUT\_x\_COMMENT** This variable holds a description or a comment for the input rule it belongs to.

Example: `PF6_INPUT_3_COMMENT='no_samba_traffic_allowed'`

**PF6\_FORWARD\_POLICY** This variable sets the default policy for packets to be forwarded by the router (FORWARD-Chain). Possible values are “REJECT” (default, rejects all packets), “DROP” (ignores all packets without further notice) and “ACCEPT” (accepts all packets). For a detailed description see the documentation of `PF_FORWARD_POLICY`.

Default setting: `PF6_FORWARD_POLICY='REJECT'`

**PF6\_FORWARD\_ACCEPT\_DEF** This variable activates the predefined rules for the FORWARD-chain of the IPv6-firewall. Possible values are “yes” and “no”.

The predefined rules open the firewall for outgoing ICMPv6-pings (one ping per second as a limit). Response packets to already allowed connections will also be allowed.

Default setting: `PF6_FORWARD_ACCEPT_DEF='yes'`

**PF6\_FORWARD\_LOG** This variable activates logging of all rejected forwarding packets. Possible values are “yes” and “no”. For a detailed description see the documentation of `PF_FORWARD_LOG`.

Default setting: `PF6_FORWARD_LOG='no'`

**PF6\_FORWARD\_LOG\_LIMIT** This variable configures the log limit for the FORWARD-chain of the IPv6-firewall to keep it readable. For a detailed description see the documentation of PF\_FORWARD\_LOG\_LIMIT.

Default setting: PF6\_FORWARD\_LOG\_LIMIT='3/minute:5'

**PF6\_FORWARD\_REJ\_LIMIT** This variable sets the limit for the rejection of forwarding TCP-packets. If a packet exceeds this limit it will be dropped without further notice (DROP). For a detailed description see the documentation of PF\_FORWARD\_REJ\_LIMIT.

Default setting: PF6\_FORWARD\_REJ\_LIMIT='1/second:5'

**PF6\_FORWARD\_UDP\_REJ\_LIMIT** This variable sets the limit for the rejection of forwarding UDP-packets. If a packet exceeds this limit it will be dropped without further notice (DROP). For a detailed description see the documentation of PF\_FORWARD\_UDP\_REJ\_LIMIT.

Default setting: PF6\_FORWARD\_UDP\_REJ\_LIMIT='1/second:5'

**PF6\_FORWARD\_N** This variable contains the number of IPv6-firewall rules for packets to be forwarded (FORWARD-chain). Two rules are activated as a default : the first denies forwarding of all local samba packets to non-local nets and the second allows this for all other local packets from the first defined IPv6-subnet.

If more local IPv6-subnets are defined the last rule has to be cloned accordingly. See also the configuration file.

Example: PF6\_FORWARD\_N='2'

**PF6\_FORWARD\_x** This variable specifies a rule for the FORWARD-chain of the IPv6-firewall. For a detailed description see the documentation of PF\_FORWARD\_x.

Differences regarding the IPv4-firewall:

- IPV6\_NET\_x has to be used instead of IP\_NET\_x.
- IPV6\_ROUTE\_x has to be used instead of IP\_ROUTE\_x.
- IPv6-addresses must be enclosed in square brackets (including the network mask, if present).
- All IPv6 address strings (including IP\_NET\_x etc.) must be enclosed in square brackets if a port or a port range follows.

Examples:

```
PF6_FORWARD_1='tmpl:samba DROP'
PF6_FORWARD_2='IPV6_NET_1 ACCEPT'
```

**PF6\_FORWARD\_x\_COMMENT** This variable holds a description or a comment for the forward rule it belongs to.

Example: PF6\_FORWARD\_1\_COMMENT='no\_samba\_traffic\_allowed'

**PF6\_OUTPUT\_POLICY** This variable sets the default strategy for outgoing packets from the router (OUTPUT chain). Possible values are "REJECT" (standard, denies all packets), "DROP" (discards all packets without further notification) and "ACCEPT" (accepts

all packages). For a more detailed description see the documentation of the Variable `PF_OUTPUT_POLICY`.

Default setting: `PF6_OUTPUT_POLICY='REJECT'`

**PF6\_OUTPUT\_ACCEPT\_DEF** This variable enables the default rules for the OUTPUT chain of the IPv6 firewall. Possible values are “yes” or “no”. Currently, there are no preset rules.

Default setting: `PF6_OUTPUT_ACCEPT_DEF='yes'`

**PF6\_OUTPUT\_LOG** This variable enables logging of all rejected outgoing packets. Possible values are “yes” or “no”. For a more detailed description see the documentation of variable `PF_OUTPUT_LOG`.

Default setting: `PF6_OUTPUT_LOG='no'`

**PF6\_OUTPUT\_LOG\_LIMIT** This variable configures the log limit for the OUTPUT chain of the IPv6 firewall, to keep the log file readable. For a more detailed description see the documentation of variable `PF_OUTPUT_LOG_LIMIT`.

Default setting: `PF6_OUTPUT_LOG_LIMIT='3/minute:5'`

**PF6\_OUTPUT\_REJ\_LIMIT** This variable configures the limit for the rejection of outgoing TCP packets. If a packet exceeds this limit the packet is discarded quietly (DROP). For a more detailed description see the documentation of variable `PF_OUTPUT_REJ_LIMIT`.

Default setting: `PF6_OUTPUT_REJ_LIMIT='1/second:5'`

**PF6\_OUTPUT\_UDP\_REJ\_LIMIT** This variable configures the limit for the rejection of outgoing UDP packets. If a packet exceeds this limit the packet is discarded quietly (DROP). For a more detailed description see the documentation of variable `PF_OUTPUT_UDP_REJ_LIMIT`.

Default setting: `PF6_OUTPUT_UDP_REJ_LIMIT='1/second:5'`

**PF6\_OUTPUT\_N** This variable contains the number of IPv6 firewall rules for incoming packets (OUTPUT chain). By default, two rules are activated: the first allows all local hosts to access the router via so-called Link-level addresses and the second allows router access for hosts from the first defined IPv6 subnet.

If several local IPv6 subnets are defined, the second rule must be added repeatedly. See the configuration file.

Example: `PF6_OUTPUT_N='1'`

**PF6\_OUTPUT\_x** This variable specifies a rule for the OUTPUT chain of the IPv6 Firewall. For a more detailed description, see the documentation of the variable `PF_OUTPUT_x`.

Differences from IPv4 firewall:

- `IPV6_NET_x` has to be used instead of `IP_NET_x`.
- `IPV6_ROUTE_x` has to be used instead of `IP_ROUTE_x`.
- IPv6 addresses must be enclosed in square brackets (including the network mask, if present).

## 1. Documentation Of Package IPV6

- All IPv6 address strings (including IP\_NET\_x etc.) must be enclosed in square brackets if followed by a port or a port range.

Examples:

```
PF6_OUTPUT_1='tmpl:ftp IPV6_NET_1 ACCEPT HELPER:ftp'
```

**PF6\_OUTPUT\_x\_COMMENT** This variable contains a description or comment to the associated OUTPUT rule.

Example: `PF6_OUTPUT_3_COMMENT='no_samba_traffic_allowed'`

**PF6\_USR\_CHAIN\_N** This variable holds the number of IPv6-firewall tables defined by the user. For a detailed description see the documentation of PF\_USR\_CHAIN\_N.

Default setting: `PF6_USR_CHAIN_N='0'`

**PF6\_USR\_CHAIN\_x\_NAME** This variable contains the name of the according user defined IPv6-Firewall table. For a detailed description see the documentation of PF\_USR\_CHAIN\_x\_NAME.

Example: `PF6_USR_CHAIN_1_NAME='usr-myvpn'`

**PF6\_USR\_CHAIN\_x\_RULE\_N** This variable contains the number of IPv6-firewall rules in the according user defined IPv6-firewall table. For a detailed description see the documentation of PF\_USR\_CHAIN\_x\_RULE\_N.

Example: `PF6_USR_CHAIN_1_RULE_N='0'`

**PF6\_USR\_CHAIN\_x\_RULE\_x** This variable specifies a rule for the user defined IPv6-firewall table. For a detailed description see the documentation of PF\_USR\_CHAIN\_x\_RULE\_x.

Differences regarding the IPv4-firewall:

- IPV6\_NET\_x has to be used instead of IP\_NET\_x.
- IPV6\_ROUTE\_x has to be used instead of IP\_ROUTE\_x.
- IPv6-addresses must be enclosed in square brackets (including the network mask, if present).
- All IPv6 address strings (including IP\_NET\_x etc.) must be enclosed in square brackets if a port or a port range follows.

**PF6\_USR\_CHAIN\_x\_RULE\_x\_COMMENT** This variable holds a description or a comment for the rule it belongs to.

Example: `PF6_USR_CHAIN_1_RULE_1_COMMENT='some_user-defined_rule'`

**PF6\_POSTROUTING\_N** This variable contains the number of IPv6 firewall rules for masking (POSTROUTING chain). For a more detailed description, see the documentation of variable PF\_POSTROUTING\_N.

Example: `PF6_POSTROUTING_N='2'`

**PF6\_POSTROUTING\_x PF6\_POSTROUTING\_x\_COMMENT**

A list of rules that describe which IPv6 packets are masked by the router (or will be forwarded unmasked). For a more detailed description see the documentation of variable PF\_POSTROUTING\_x.

## 1. Documentation Of Package IPV6

**PF6\_PREROUTING\_N** This variable contains the number of IPv6 firewall rules for forwarding to a different destination (PREROUTING chain). For a more detailed description see the documentation of variable PF\_PREROUTING\_N.

Example: PF6\_PREROUTING\_N='2'

### **PF6\_PREROUTING\_x PF6\_PREROUTING\_x\_COMMENT**

A list of rules to set the IPv6 packets that should be forwarded by the router to another destination. For a more detailed description see the documentation of variable PF\_PREROUTING\_x.

#### **1.1.4. Web-GUI**

The package installs a menu entry “Packet Filter (IPv6)” in Mini-HTTPD to review entries of the packet filter configured for IPv6 .

# A. Appendix Of Package IPV6

## A.1. IPV6 - Connection to IPv6-Internet using a SixXS-Tunnel

This section describes how the package IPV6 can be used to connect your home network to the IPv6 Internet by using a tunnel from the provider SixXS (<https://www.sixxs.net/>).

### A.1.1. Get An Account

First, apply for a SixXS account under “Signup for new users”. After that you have a user name in the form YYYYY-Sixxs and an associated password. This data is needed later for the tunnel configuration.

### A.1.2. Tunnel Configuration

#### Preparation

At first you have to apply for the tunnel. This happens after registration via the menu item “request tunnel”. It is important to select “Dynamic IPv4 Endpoint using Heartbeat protocol” as the type of the tunnel in the second entry because this configuration is supported directly by the fli4l. The third variant “Static IPv4 Endpoint” is also possible if you own a dedicated IPv4 address that never changes. The tunnel variant “Dynamic NAT traversing IPv4 endpoint is using AYIYA” currently is not supported by the IPV6 package.

Once you’ve filled in the details for the location of the router into the other fields the next step is going to the second page via “changed”. Here one or multiple PoPs (Points of Presence) have to be chosen. They are important for the tunnel construction later. You should take the one which is closest to you in order to make tunneling of IPv6 packets as efficiently as possible.

If all details are filled in and sent via “Place request for new tunnel” you will receive an e-mail with the necessary tunnel data. This includes:

1. Identification number of the tunnel (T...)
2. Name of the associated PoP
3. IPv4-address of the associated PoP (“SixXS IPv4”)
4. Local IPv6-address and subnet mask of the tunnel (typically /64 for SixXS) - this represents the address of your router (“Your IPv6”)
5. Remote IPv6 address of the tunnel including subnet mask (identical with the subnet mask of the local IPv6 address), ie the address of the (PoPs SixXS “ IPv6”)



## Configuration

Now the tunnel can be configured! First, the variable `IPV6_TUNNEL_N` has to be set to “1” because exactly one tunnel should be built:

```
IPV6_TUNNEL_N='1'
```

The details SixXS provided have to be filled in the IPv6 configuration as follows:

1. The identification number of the tunnel is put in `IPV6_TUNNEL_1_TUNNELID`.
2. The name of the associated PoP is not needed
3. The IPv4-address of the associated PoP has to be specified in `IPV6_TUNNEL_1_REMOTEV4`.
4. The local IPv6-address *and* subnet mask of the tunnel is filled in the variable `IPV6_TUNNEL_1_LOCALV6`.
5. The remote IPv6 address of the tunnel *without* subnet mask goes to variable `IPV6_TUNNEL_1_REMOTEV6`.

In addition the username and password have to be specified in the tunnel configuration in variables `IPV6_TUNNEL_1_USERID` and `IPV6_TUNNEL_1_PASSWORD`.

Finally set the variable `IPV6_TUNNEL_1_TYPE` to reflect that the configured tunnel is a SixXS tunnel:

```
IPV6_TUNNEL_1_TYPE='sixxs'
```

Example: If you got the PoP “deham01” with the IPv4-address 212.224.0.188 and tunnel end points 2001:db8:900:551::1/64 (remote) and 2001:db8:900:551::2/64 (local) from SixXS, the tunnel-ID is “T1234”, the username “USER1-SIXXS” and the password “sixxs” (do *not* use this password!) then the resulting configuration will look like this:

```
IPV6_TUNNEL_N='1'
IPV6_TUNNEL_1_LOCALV4='dynamic' # or fixed local IPv4-address
IPV6_TUNNEL_1_REMOTEV4='212.224.0.188'
IPV6_TUNNEL_1_LOCALV6='2001:db8:900:551::2/64'
IPV6_TUNNEL_1_REMOTEV6='2001:db8:900:551::1'
IPV6_TUNNEL_1_TYPE='sixxs'
IPV6_TUNNEL_1_USERID='USER1-SIXXS'
IPV6_TUNNEL_1_PASSWORD='sixxs'
IPV6_TUNNEL_1_TUNNELID='T1234'
```

## Test

If you accomplished this then update and restart the `fli4l`. After logging in to the router (directly or e.g. via SSH) you should already be able to ping the tunnel endpoint. With the example data above this will look as follows:

## A. Appendix Of Package IPV6

```
garm 3.6.0-revXXXXX # ping -c 4 2001:db8:900:551:0:0:0:1
PING 2001:db8:900:551::1 (2001:db8:900:551::1): 56 data bytes
64 bytes from 2001:db8:900:551::1: seq=0 ttl=64 time=67.646 ms
64 bytes from 2001:db8:900:551::1: seq=1 ttl=64 time=72.001 ms
64 bytes from 2001:db8:900:551::1: seq=2 ttl=64 time=70.082 ms
64 bytes from 2001:db8:900:551::1: seq=3 ttl=64 time=67.996 ms

--- 2001:db8:900:551::1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 67.646/69.431/72.001 ms
```

Important is the part with the “0 % packet loss”. This means that response packets have been received for all PING packets. If you get no response from the other end of the tunnel, the result is different:

```
garm 3.6.0-revXXXXX # ping -c 4 2001:db8:900:551:0:0:0:1
PING 2001:db8:900:551::1 (2001:db8:900:551::1): 56 data bytes

--- 2001:db8:900:551::1 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
```

You may notice that not for a single ping a response packet was received (“100% packet loss”). This either means that the configuration is incorrect or that the tunnel has not been established fully by SixXS yet. In the second case you should wait for some time because the configuration on the PoPs may last a few hours. If you double-checked the configuration and discovered no mistakes, and some time has elapsed without the tunnel working, you should contact SixXS by e-mail and describe the problem in detail.

### A.1.3. Configuration Of The Subnet

#### Preparations

If the tunnel is working you made the first major step. But you have not yet finished. For now only the router has the ability to send and receive packets to and from the IPv6 internet. But the hosts in the local net can’t do this by now. An IPv6 subnet has to be configured to which the local hosts are bound.

Here a small but significant difference to the configuration of an IPv4 network is to be noticed: Because of the address shortage usually only one host is connected directly to the internet. The other hosts on the local network receive only internal network IP addresses that are not routed to the outside. These are in the ranges 192.168.\*.\*, 172.16.\*.\* to 172.31.\*.\*, and 10.\*.\*.\*, depending on the size of the subnet. <sup>1</sup>

With IPv6 we get far enough IP addresses thus eliminating the need to use internal network addresses. Due to the global nature of local subnets it must be assured that the addresses of local hosts do not collide with other addresses on the Internet. Therefore a subnet of the IPv6 Provider has to be allocated in order to avoid such collisions.

SixXS does this with the menu item “Request subnet”. Here you mainly have to specify the tunnel to be used. This is easy because only one tunnel has been configured so far. After submitting the form via “Place request for new subnet” you will, after some time, obtain the following information by e-mail:

---

<sup>1</sup>see RFC 1918(<http://tools.ietf.org/html/rfc1918>) for details

1. IPv6-address and subnet mask of the subnet (“Subnet IPv6”)
2. IPv6-address of the routers in the tunnel to where the subnet will be routed by SixXS (“Routed to”)
3. IPv4-address of the router (“Your IPv4”)<sup>2</sup>

This data is sufficient for configuring an own IPv6 subnet with fli4l. One more thing must be known: The assigned subnet is usually very large. SixXS usually allocates /48-subnets, i.e. within the 128-bit IPv6 address the proportion of the network prefix is 48 bits and the proportion that is available for addressing of hosts is  $128 - 48 = 80$  bits. Such a large subnet has two major drawbacks. The first disadvantage is the sheer size: this network can address  $2^{80} \approx 1209$  trillion hosts. It appears inadvisable to use this without structuring the host portion of the address. The second drawback is more serious: within such a large subnet the so-called *IPv6-autoconfiguration* does not work. This is a process in which the IPv6 host receives the subnet prefix on defined protocols and derives its IPv6 address from the MAC address of its network adapter. The MAC address consists of six bytes. Using the standard EUI-64 you can stretch it to eight bytes. This corresponds to 64 bits in the end. For 80-bit simply not enough information is available on the host.

Long story short: The subnet must be made smaller. It has to become a /64 subnet for auto-configuration to work properly. But that’s easy: the subnet mask has to be changed to /64. If SixXS e.g. assigned the subnet `2001:db8:123::/48` then the subnet for fli4l is just set to `2001:db8:123::/64`. In detail this means that the /48 subnet is divided in  $2^{(64-48)} = 2^{16} = 65536$  sub-subnets. The first with the number zero is to be used with fli4l. You have to remember that the short form `2001:db8:123::` really represents the long address `2001:db8:123:0:0:0:0:0`. The first three numbers are the IPv6 provider’s globally unique part of the subnet, the fourth number represents the selected sub-subnet “zero”,<sup>3</sup> and the last four numbers are reserved for the host portion. This still gives a huge (sub-) subnet where up to  $2^{64} \approx 18,4$  trillion hosts can be accommodated. Thanks to the IPv6 autoconfiguration you will not have to bother about the actual addresses. And that’s good ...

### Configuration

Back to the configuration! At first the variable `IPV6_NET_N` is set to “1” because exactly one local IPv6 subnet has to be established. The IPv6 address of the /64 subnet including the subnet mask goes to the variable `IPV6_NET_1`. But that’s not completely right: here the IPv6 address of *the router within that subnet* is to be set, but *without* the subnet prefix that is associated with the tunnel. In fact this is configured somewhere else: in the tunnel configuration. There the variable `IPV6_TUNNEL_1_PREFIX` has to be set to the requested subnet prefix.

Example: Having received the /48er-IPv6-subnet `2001:db8:123::/48` from SixXS, chosen it’s subnet with the number ‘456’ to be used as a /64 sub-subnet and finally determined that the router within that subnet should get the address of “1”, we get the following configuration:

```
IPV6_NET_N='1'
IPV6_NET_1='0:0:0:456::1/64'           # IPv6-address of the routers (without
                                       # subnet-prefix) + subnet mask
IPV6_TUNNEL_1_PREFIX='2001:db8:123::/48' # /48-subnet-prefix
```

<sup>2</sup>in case that the router gets its IPv4-address dynamically this will specify “heartbeat”

<sup>3</sup>Of course you can have another sub-subnet!

## A. Appendix Of Package IPV6

It should be noted that the first three zeros in IPV6\_NET\_1 hold the place for the /48-subnet-prefix associated with the tunnel. Together with the /48 subnet prefix assigned by the tunnel provider this results in the /64-Subnetz 2001:db8:123:456::/64 and the IPv6 router address 2001:db8:123:456::1.

Now we need the name of the network interface to which this subnet has to be bound. Each subnet is bound to exactly one network interface. If only one configured network card is present in the router the name of the network interface is typically “eth0” for wired or “wlan0” for wireless adapters. If in doubt have a look at IP\_NET\_1\_DEV (“IP” without “6”) and copy the content:

```
IPV6_NET_1_DEV='eth0' # Network interface for this IPv6-subnet
```

Finally we just need to let IPv6 autoconfiguration do the rest:

```
IPV6_NET_1_ADVERTISE='yes'      # /64-subnet-prefix and default-route per RA
IPV6_NET_1_ADVERTISE_DNS='yes'  # DNS-server per RA (needs
                                # DNS_SUPPORT_IPV6='yes'!)
IPV6_NET_1_DHCP='yes'           # Domain-name and DNS-server per DHCPv6
                                # (needs DNS_SUPPORT_IPV6='yes')
```

The last two settings are not absolutely necessary for a working IPv6 subnet but are very helpful. They serve to spread additional information on the IPv6 subnet, i.e. IPv6 address of the DNS server and the domain used. The DNS server even may be published in two ways. Because different systems have different preferences it is of advantage to activate both methods (RDNSS via router advertisements and DHCPv6).

### Test

The whole IPv6-configuration of this example (DNS\_SUPPORT\_IPV6='yes' is assumed!) looks like this:

```
IPV6_NET_N='1'
IPV6_NET_1='0:0:0:456::1/64'  # IPv6-address of the routers (without
                                # Subnet-prefix) + subnet mask
IPV6_NET_1_DEV='eth0'         # network interface for this IPv6-subnet
IPV6_NET_1_ADVERTISE='yes'   # /64-subnetz-prefix and default-route per RA
IPV6_NET_1_ADVERTISE_DNS='yes' # DNS-server per RA
IPV6_NET_1_DHCP='yes'        # Domain-name and DNS-server per DHCPv6

IPV6_TUNNEL_N='1'
IPV6_TUNNEL_1_PREFIX='2001:db8:123::/48' # /48-subnet mask
IPV6_TUNNEL_1_LOCALV4='dynamic' # or fixed local IPv4-address
IPV6_TUNNEL_1_REMOTEV4='212.224.0.188'
IPV6_TUNNEL_1_LOCALV6='2001:db8:900:551::2/64'
IPV6_TUNNEL_1_REMOTEV6='2001:db8:900:551::1'
IPV6_TUNNEL_1_TYPE='sixxs'
IPV6_TUNNEL_1_USERID='USER1-SIXXS'
IPV6_TUNNEL_1_PASSWORD='sixxs'
IPV6_TUNNEL_1_TUNNELID='T1234'
```

## A. Appendix Of Package IPV6

With a fli4l like this a normally configured Windows 7 host should automatically configure its IPv6 address, default route, DNS server and domain and thus make the computer IPv6 capable. This can be verified e.g. with a simple PING from the Windows host to the IPv6 internet. In the following example we try to reach the fli4l.de webserver from the Windows PC (we directly use the IPv6 address here to avoid assumption of correct DNS functionality):

```
C:\>ping 2001:bf0:c000:a::2:132
```

```
Ping executed for 2001:bf0:c000:a::2:132 with 32 bytes data:
```

```
Answer from 2001:bf0:c000:a::2:132: time=104ms
Answer from 2001:bf0:c000:a::2:132: time=102ms
Answer from 2001:bf0:c000:a::2:132: time=106ms
Answer from 2001:bf0:c000:a::2:132: time=106ms
```

```
Ping-Statistics for 2001:bf0:c000:a::2:132:
```

```
Packets: Sent = 4, Received = 4, lost = 0 (0% lost),
time in millisecs.:
Minimum = 102ms, Maximum = 106ms, Average = 104ms
```

Yo can try to use the tool “tracert” (in Windows: “tracert”) in order to examine whether a packet is routed correctly. An example from the local network of the author is found below. This allows to notice that a packet first reaches fli4l (first line), then the other end of the tunnel (second row) and finally the global IPv6 internet (from the third line):

```
C:\>tracert 2001:bf0:c000:a::2:132
```

```
Route tracing to virtualhost.in-berlin.de [2001:bf0:c000:a::2:132]
on a maximum of 30 hops:
```

1	<1 ms	<1 ms	<1 ms	garm.example.org [2001:db8:13da:1::1]
2	70 ms	79 ms	71 ms	gw-1362.ham-01.de.sixxs.net [2001:db8:900:551::1]
3	67 ms	71 ms	76 ms	2001:db8:800:1003::209:55
4	68 ms	*	70 ms	2001:db8:1:0:87:86:71:240
5	69 ms	*	71 ms	2001:db8:1:0:87:86:77:67
6	72 ms	*	71 ms	2001:db8:1:0:86:87:77:81
7	71 ms	*	71 ms	2001:db8:1:0:87:86:77:83
8	90 ms	*	81 ms	2001:db8:1:0:87:86:77:62
9	84 ms	*	88 ms	2001:db8:1:0:87:86:77:71
10	99 ms	83 ms	83 ms	2001:db8:1:0:87:86:77:249
11	94 ms	87 ms	87 ms	20gigabitethernet4-3.core1.fra1.he.net [2001:7f8::1b1b:0:1]
12	96 ms	99 ms	99 ms	10gigabitethernet1-4.core1.ams1.he.net [2001:470:0:47::1]
13	105 ms	108 ms	107 ms	2001:7f8:8:5:0:73e6:0:1
14	106 ms	107 ms	104 ms	virtualhost.in-berlin.de [2001:bf0:c000:a::2:132]

```
Tracing ended.
```

## List of Figures

# List of Tables

# Index

HOSTNAME\_IP6, [4](#)

IPV6\_NET\_N, [4](#)

IPV6\_NET\_x, [4](#)

IPV6\_NET\_x\_ADVERTISE, [5](#)

IPV6\_NET\_x\_ADVERTISE\_DNS, [6](#)

IPV6\_NET\_x\_DEV, [5](#)

IPV6\_NET\_x\_DHCP, [6](#)

IPV6\_NET\_x\_NAME, [6](#)

IPV6\_NET\_x\_TUNNEL, [5](#)

IPV6\_ROUTE\_N, [9](#)

IPV6\_ROUTE\_x, [9](#)

IPV6\_TUNNEL\_N, [6](#)

IPV6\_TUNNEL\_x\_DEFAULT, [7](#)

IPV6\_TUNNEL\_x\_DEV, [8](#)

IPV6\_TUNNEL\_x\_LOCALV4, [7](#)

IPV6\_TUNNEL\_x\_LOCALV6, [8](#)

IPV6\_TUNNEL\_x\_MTU, [8](#)

IPV6\_TUNNEL\_x\_PASSWORD, [8](#)

IPV6\_TUNNEL\_x\_PREFIX, [7](#)

IPV6\_TUNNEL\_x\_REMOTEV4, [7](#)

IPV6\_TUNNEL\_x\_REMOTEV6, [8](#)

IPV6\_TUNNEL\_x\_TIMEOUT, [9](#)

IPV6\_TUNNEL\_x\_TUNNELID, [9](#)

IPV6\_TUNNEL\_x\_TYPE, [6](#)

IPV6\_TUNNEL\_x\_USERID, [8](#)

OPT\_IPV6, [4](#)

PF6\_FORWARD\_ACCEPT\_DEF, [11](#)

PF6\_FORWARD\_LOG, [11](#)

PF6\_FORWARD\_LOG\_LIMIT, [11](#)

PF6\_FORWARD\_N, [12](#)

PF6\_FORWARD\_POLICY, [11](#)

PF6\_FORWARD\_REJ\_LIMIT, [12](#)

PF6\_FORWARD\_UDP\_REJ\_LIMIT, [12](#)

PF6\_FORWARD\_x, [12](#)

PF6\_FORWARD\_x\_COMMENT, [12](#)

PF6\_INPUT\_ACCEPT\_DEF, [10](#)

PF6\_INPUT\_ICMP\_ECHO\_REQ\_LIMIT, [10](#)

PF6\_INPUT\_ICMP\_ECHO\_REQ\_SIZE, [10](#)

PF6\_INPUT\_LOG, [10](#)

PF6\_INPUT\_LOG\_LIMIT, [10](#)

PF6\_INPUT\_N, [10](#)

PF6\_INPUT\_POLICY, [9](#)

PF6\_INPUT\_REJ\_LIMIT, [10](#)

PF6\_INPUT\_UDP\_REJ\_LIMIT, [10](#)

PF6\_INPUT\_x, [11](#)

PF6\_INPUT\_x\_COMMENT, [11](#)

PF6\_LOG\_LEVEL, [9](#)

PF6\_OUTPUT\_ACCEPT\_DEF, [13](#)

PF6\_OUTPUT\_LOG, [13](#)

PF6\_OUTPUT\_LOG\_LIMIT, [13](#)

PF6\_OUTPUT\_N, [13](#)

PF6\_OUTPUT\_POLICY, [12](#)

PF6\_OUTPUT\_REJ\_LIMIT, [13](#)

PF6\_OUTPUT\_UDP\_REJ\_LIMIT, [13](#)

PF6\_OUTPUT\_x, [13](#)

PF6\_OUTPUT\_x\_COMMENT, [14](#)

PF6\_POSTROUTING\_N, [14](#)

PF6\_POSTROUTING\_x, [14](#)

PF6\_POSTROUTING\_x\_COMMENT, [14](#)

PF6\_PREROUTING\_N, [14](#)

PF6\_PREROUTING\_x, [15](#)

PF6\_PREROUTING\_x\_COMMENT, [15](#)

PF6\_USR\_CHAIN\_N, [14](#)

PF6\_USR\_CHAIN\_x\_NAME, [14](#)

PF6\_USR\_CHAIN\_x\_RULE\_N, [14](#)

PF6\_USR\_CHAIN\_x\_RULE\_x, [14](#)

PF6\_USR\_CHAIN\_x\_RULE\_x\_COMMENT, [14](#)