

Package PROXY

Version 4.0.0-trunk-x86_64-r60718

Frank Meyer
email: frank@fli4l.de

The fli4l-Team
email: team@fli4l.de

August 23, 2022

Contents

1	Documentation For Package PROXY	3
1.1	PROXY - Several Proxy Servers	3
1.1.1	OPT_PRIVOX - A HTTP-Proxy Not Only For Ad Filtering	3
1.1.2	OPT_TOR - An Anonymous Communication System For The Internet	5
1.1.3	OPT_SS5 - Ein Socks4/5 Proxy	6
1.1.4	OPT_TRANSPROXY (EXPERIMENTAL) - Transparent HTTP Proxy	7
1.1.5	OPT_SIPROXD (experimental) - Proxy for Session Initiation Protocol	7
1.1.6	OPT_IGMPPROXY - Internet Group Management Protocol Proxy) . .	8
1.1.7	OPT_STUNNEL - Tunneling Connections Over SSL/TLS	14
	List of Figures	21
	List of Tables	22
	Index	23

1 Documentation For Package PROXY

1.1 PROXY - Several Proxy Servers

1.1.1 OPT_PRIVoxy - A HTTP-Proxy Not Only For Ad Filtering

The Privoxy homepage (<http://www.privoxy.org/>) qualifies it as a “Privacy Enhancing Proxy”. As a side-effect it replaces ad banners and popups with empty pictures, prevents saving of unwanted cookies (small files which make it possible for websites to track users) and so-called web-bugs (1x1 pixel sized pictures that are also used to track user behavior).

Privoxy can be configured and deactivated via a web interface (as long as its running). This web interface is found at <http://config.privoxy.org/> or <http://p.p/>. This configuration does not survive reboots...

Privoxy is an enhancement of Internet Junkbuster which has been contained in this package till version 2.1.0 (<http://www.junkbuster.com/>). All filtering rules are defined in a central file `default.action`. On fli4l it can be found in the directory `/etc/privoxy`. The main advantage of this method is that new versions can be downloaded separately at <http://sourceforge.net/projects/ijbswa/files/>. Each user of fli4l can keep the file up-to-date in this way without the need for fli4l updates.

PRIVOXY_MENU Adds a privoxy entry to the httpd menu.

PRIVOXY_N Sets the number of privoxy instances that should be started.

PRIVOXY_x_LISTEN Specify IP addresses or symbolic names including portnumber of the interface here on which Privoxy should listen to clients. It is a good idea to specify only trusted interfaces because all clients have full access to privoxy (and its activated configuration editor). Normally setting `IP_NET_1_IPADDR:8118` makes most sense.

Privoxy will listen to the addresses set here offering its services. The default port is 8118. This setting has to be used in the proxy configuration of your Internet browser. Additional informations on client configuration can be found at <http://www.privoxy.org/>

Define your fli4l name (see `HOSTNAME` in `base.txt`) or its IP (i.e. 192.168.6.1) as a proxy in your client. With the port number set here all data necessary to configure a web browser for using privoxy is provided.

PRIVOXY_x_ALLOW_N Set the number of list entries.

PRIVOXY_x_ALLOW_x List of nets and/or IP addresses for which the packet filter has to be opened. Default: `IP_NET_1`.

PRIVOXY_x_ACTIONDIR This variable sets the path to the privoxy rulesets (*default.action* and *user.action*) on the router. This variable can be used for two things:

Moving rulesets to permanent storage If you specify a path outside of the RAM disk standard rulesets will be copied there on first boot and further on the new path is used. Changes to rulesets will survive reboots then. Please note that changed rulesets due to privoxy updates will be ignored.

Use of own rulesets fli4l allows standard rulesets to be overwritten by user defined rulesets. Create a subdirectory (i.e. *etc/my_privoxy* - but not *etc/privoxy!*) in the *config* directory to place your own rulesets there.

Setting this variable is optional.

PRIVOXY_x_HTTP_PROXY If an additional http proxy should be used in conjunction with privoxy (i.e. as a webcache) it can be specified here. Privoxy will use this proxy then. An entry could be like this:

```
PRIVOXY_1_HTTP_PROXY='my.provider.de:8000'
```

This setting is optional.

PRIVOXY_x SOCKS_PROXY If another SOCKS proxy should be used in addition to Privoxy it can be specified here. To ensure privacy data traffic may be i.e. sent through the Tor network. For details see the [Tor Documentation](#) (Page 5). An entry for using Tor could look like this:

```
PRIVOXY_x SOCKS_PROXY='127.0.0.1:9050'
```

This setting is optional.

PRIVOXY_x_TOGGLE This option activates toggling the proxy over the web interface. If Privoxy is switched off it will act as a simple forwarding proxy and won't change page content in any way. Please note that this setting affects ALL proxy users (if one user disables privoxy it acts as a forwarding proxy for all users).

PRIVOXY_x_CONFIG This option enables interactive configuration editing for proxy users using Privoxy's web interface. For further details please consult the Privoxy documentation.

PRIVOXY_x_LOGDIR This option specifies a log directory for Privoxy. This may be useful for i.e. logging user accesses to websites. If nothing is set here only important messages will be logged to console and **PRIVOXY_LOGLEVEL** is ignored.

You may specify 'auto' to make fli4l use the path of the system directory for persistent storage. Please take care for **FLI4L_UUID** being correctly configured as huge data amounts should be expected and /boot or even RAM disk may else overflow.

PRIVOXY_x_LOGLEVEL This option specifies what kind of informations Privoxy should log. The following values are possible (they may be divided by spaces or simply added):

Value	What will be logged?
1	Each request (GET/POST/CONNECT)
2	Status of each connection
4	show I/O status
8	show header parsing
16	log all data
32	debug force feature
64	debug regular expression
128	debug fast forwarding
256	debug GIF de-animation
512	common log format (for logfile analysis)
1024	debug Popup-Kill function
2048	log access to the builtin web server
4096	Start messages and warnings
8192	Non-fatal errors

To create a log file in common logfile format ONLY 512 should be set otherwise the logfile would be 'polluted' by other messages that prevent proper analysis.

Privoxy offers lots of configuration options. Not all can be offered in fli4l's config files. Many of these options can be set through Privoxy's web interface. More informations on configuration files can be found on Privoxy's homepage. The configuration files are found at `<fli4l-directory>/opt/etc/privoxy/`. These are original files from Privoxy packages with all comments removed because of space limitations.

1.1.2 OPT_TOR - An Anonymous Communication System For The Internet

Tor is a tool for a lot of organisations and humans that want to improve their protection and security while using the Internet. Using Tor aids them to anonymise browsing, publishing to the web, instant messaging, IRC, SSH and other TCP based services. Tor also is a platform for software developers to create new tools for enhanced anonymity, security and privacy protection.

<https://www.torproject.org/index.html.de>

TOR_LISTEN_N

TOR_LISTEN_x Specify IP addresses or symbolic names including portnumber of the interface here on which Tor should listen to clients. It is a good idea to specify only trusted interfaces because all clients have full access to Tor (and its activated configuration editor). Normally setting `IP_NET_1_IPADDR:9050` makes most sense.

Tor will listen to the addresses set here offering its services. The default port is 9050. This setting has to be used in the configuration of your programs.

Define your fli4l name (see `HOSTNAME` in `base.txt`) or its IP (i.e. 192.168.6.1) as a proxy in your client. With the port number set here all data necessary to configure programs for using Tor is provided.

TOR_ALLOW_N Sets the number of list entries.

TOR_ALLOW_x List of nets and/or IP addresses for which the packet filter has to be opened.
Default: IP_NET_1.

TOR_CONTROL_PORT Specify here on which TCP port Tor should open a control port using Tor control protocol. The setting is optional. If nothing is specified this function will be deactivated.

TOR_CONTROL_PASSWORD Set a password for the control port here.

TOR_DATA_DIR This setting is optional. If not set the default directory /etc/tor will be used.

TOR_HTTP_PROXY If Tor should forward queries to a HTTP proxy set it here. Tor will use the proxy then to use its functions. A valid entry could look like this:

```
TOR_HTTP_PROXY='my.provider.de:8000'
```

This setting is optional.

TOR_HTTP_PROXY_AUTH If the proxy needs authentication set it here. Use notation username:password.

TOR_HTTPS_PROXY A HTTPS proxy can be specified here. See [TOR_HTTP_PROXY](#).

TOR_HTTPS_PROXY_AUTH See [TOR_HTTP_PROXY_AUTH](#).

TOR_LOGLEVEL This option sets Tor's logging behavior. Possible values are:

debug, info, notice, warn or err.

Don't use debug and info due to security concerns except when really needed.

TOR_LOGFILE If Tor should write its log to a file instead of syslog you can set its name here.

You may specify 'auto' to make fli4l use the path of the system directory for persistent storage. Please take care for FLI4L_UUID being correctly configured as huge data amounts should be expected and /boot or even RAM disk may else overflow.

1.1.3 OPT_SS5 - Ein Socks4/5 Proxy

For some programs a Socks proxy may be needed. SS5 provides this functionality.

<http://ss5.sourceforge.net/>

SS5_LISTEN_N

SS5_LISTEN_x Specify IP addresses or symbolic names including portnumber of the interface here on which SS5 should listen to clients. It is a good idea to specify only trusted interfaces because all clients have full access to SS5 (and its activated configuration editor). Normally setting IP_NET_1_IPADDR:8050 makes most sense.

SS5 will listen to the addresses set here offering its services. The default port is 8050. This setting has to be used in the configuration of your programs.

Define your fli4l name (see HOSTNAME in base.txt) or its IP (i.e. 192.168.6.1) as a proxy in your client. With the port number set here all data necessary to configure programs for using SS5 is provided.

SS5_ALLOW_N Sets the number of list entries.

SS5_ALLOW_x List of nets and/or IP addresses for which the packet filter has to be opened.
Default: IP_NET_1.

1.1.4 OPT_TRANSPROXY (EXPERIMENTAL) - Transparent HTTP Proxy

Transproxy is a „transparent” Proxy - a program that catches all HTTP requests going through the fli4l router and redirects them to a normal HTTP proxy i.e. Privoxy. To achieve this the packet filters has to redirect HTTP queries that should go to the Internet to Transproxy which will then redirect them to another HTTP proxy. It uses iptables's „REDIRECT” function to accomplish this:

```
PF_PREROUTING_1='tpr: http IP_NET_1 REDIRECT:8081'
```

This rule would redirect all HTTP packets from the first defined net (internal LAN normally) to Transproxy on port 8081.

TRANSPROXY_LISTEN_N

TRANSPROXY_LISTEN_x Specify IP addresses or symbolic names including portnumber of the interface here on which Transproxy should listen to clients. All interfaces have to be specified here that should redirect their packets to Transproxy by the packet filter. With the default setting `any:8081` Transproxy listens on all interfaces.

TRANSPROXY_TARGET_IP

TRANSPROXY_TARGET_PORT With this options it is set to which service incoming HTTP queries should be redirected. This can be a standard HTTP proxy (Squid, Privoxy, Apache, a.s.o.) on a random PC (or fli4l itself). Please ensure that this proxy is not in the range of the HTTP queries redirected by the packet filter. This would cause an infinite loop otherwise.

TRANSPROXY_ALLOW_N

TRANSPROXY_ALLOW_x List of nets and/or IP addresses for which the packet filter has to be opened. It should cover the nets that should be redirected by the packet filter. If you don't set any ranges here they have to be entered manually in the configuration of the packet filter.

1.1.5 OPT_SIPROXD (experimental) - Proxy for Session Initiation Protocol

If SIP tools (Ekiga, x-lite or Hardware SIP-Phones) have to be used behind a router it may be needed that network ports are redirected for connections to work properly.

To avoid this a special SIP Proxy-Server can be used. At the time of writing (fli4l V4.0.0) several such Proxys are being evaluated. If someone can suggest a good choice please inform us!

1.1.6 OPT_IGMPProxy - Internet Group Management Protocol Proxy)

The German Telekom AG offers VDSL25/50 for some years now (Bandwidth: 25/50 MBit/s) in so-called Entertain packages. This enables to watch TV over the Internet (IPTV).

IPTV is provided via Multicast, from one source to a (closed) group. The network protocol needed to organize Multicast groups is called IGMP (Internet Group Management Protocol). IGMP (<http://en.wikipedia.org/wiki/IGMP>) provides the ability to dynamically manage Multicast groups. The administration is not done in the transmitting station but by the routers to which recipients of a multicast group are connected directly. IGMP provides functions by which a station notifies a router that it wants to receive multicast IP packets of a particular multicast group.

The provided Speedport routers (W700V/W701V/W722 by the time of writing) support IGMP. Those who want to use fli4l for IPTV instead of those Speedport routers, need an IGMP-Proxy (<http://sourceforge.net/projects/igmpproxy/>) on the fli4l. OPT_IGMPProxy is a IGMP-Proxy for fli4l.

This documentation for the package OPT_IGMP describes the configuration of fli4l to use VDSL and IPTV with the supplied set-top box (STB) X300T/X301T or MR-303 behind a fli4l router. In this description, the installation of IPTV via an additional network card is assumed.

Precondition

The German Telekom introduced VDSL as a VLAN. In the introductory phase only one VLAN tag (ID7) was used for all traffic. After that they switched to two VLAN tags (ID7, ID8). The Internet traffic remains on ID7 and the new ID8 is used exclusively for IPTV multicast traffic. The conversion of VDSL (two VLAN tags ID7/ID8) is largely completed by the current state.

Hardware (besides Set-Top-Boxes and VDSL-Modem):

- Hardware for fli4l: For VDSL 25/50 more than an i486. In case of sound or image distortions it may be that the hardware used has too little power.
- High-End network interface cards (Examples: 3Com, Intel Pro100). Realtek chipsets are not recommended.

Software:

- Package: `advanced_networking`
- Package: `dhcp_client` (for the use of ID8)

The following describes adapting the config files `base.txt`, `dsl.txt`, `advanced_networking.txt`, `dhcp_client.txt`, `dns_dhcp.txt`.

Hardware Setup

The recommendation to connect the IPTV SetTopBox without further network elements directly to the router also applies to fli4l. If network nodes like hubs, switches, bridges, gateways, or routers have to be placed between the IPTV box and fli4l, these should be multicast-enabled to avoid problems.

Home networks usually do not use switches that separate virtual networks (VLAN) from each other in order to disencumber remaining traffic (ID7) from IPTV multicast traffic (ID8).

This is why a separate NIC (Network Interface Card = LAN or Ethernet card) is used to connect the set-top box (STB) directly to fli4l to avoid all problems due to heavy traffic. Those preferring the single NIC method (not described here) should know what to do on their own.

The easiest way to use an IPTV STB hence is to install an additional NIC to fli4l's hardware. Find a diagram below how to migrate fli4l from standard to a third NIC:

- Standard configuration:
 - eth0 is used as the NIC for internal home/office LAN in base.txt
 - eth1 is the DSL interface mentioned in dsl.txt



Figure 1.1: fli4l in a standard configuration

- Advanced configuration with an additional IPTV NIC:
 - After installation of an additional NIC eth2 has to be inserted in base.txt.



Figure 1.2: fli4l in an IPTV configuration

VLAN Configuration

First of all: OPT_IGMP is not depending on VLAN. VLAN is needed for Deutsche Telekom VDSL and hence should be supported by the router. Whether VLAN is required for other providers (Arcor, Alice, a.s.o. ..) is beyond our knowledge at the moment.

To get VDSL25/50 as provided by T-Home up and running, the NIC to the VDSL modem necessarily has to be configured as a VLAN interface.

A note for those using only 'normal DSL', ie ADSL, ADSL2, ADSL2+: VLAN is only needed by VDSL, but not for 'normal DSL' - the VLAN configuration will not work in this case.

If two VLAN tags are used (see above) traffic is split as follows:

- VLAN ID7: Internet traffic
- VLAN ID8: IPTV Multicast traffic

This way Internet traffic is independent from IPTV traffic. The main difference is that VLAN ID7 needs a PPPoE dial-in. VLAN ID8 is provided via a DHCP server without dial-in. In this architecture there is no forced disconnect after 24 hours any more.

The following configuration is needed for VLAN (hardware setup of NICs as described above):

advanced_networking.txt

```
VLAN_DEV_N='2'
VLAN_DEV_1_DEV='eth1'      # interface of VDSL-Modem; example: eth1
                           # in our example 'eth1' connects to the VDSL modem
VLAN_DEV_1_VID='7'         # ID7 to support VLAN for internet
VLAN_DEV_2_DEV='eth1'      # interface of VDSL modem; example: eth1
VLAN_DEV_2_VID='8'         # ID8 to support VLAN for IPTV
```

The virtual NIC eth1.7 has to be inserted into the DSL configuration:

dsl.txt

```
PPPOE_ETH='eth1.7'        # eth<number of the card connecting the vdsl modem>.7'
                           # i.e. 'eth1.7'
```

The virtual NIC eth1.8 needs a dhcp_client, because VLAN ID8 is provided by a DHCP server without dial-in.

dhcp_client

```
OPT_DHCP_CLIENT='yes'
DHCP_CLIENT_TYPE='dhcpcd'
DHCP_CLIENT_INTERFACES='IP_NET_3_DEV' # listen on interface eth1.8
DHCP_CLIENT_USEPEERDNS='no'
DHCP_CLIENT_HOSTNAME=''
```

As of fli4l V3.3 the interface can only be defined by the value of IP_NET_x_DEV defined for the interface in base.txt, here: IP_NET_3_DEV. Specifying eth1.8 is not possible anymore.

Optional:

If the NIC in use has problems with the MTU size it can be adapted with the parameter DEV_MTU. Intel Pro/100 (e100) and a 3-Com card worked well during tests without changes, but a 3Com '3c59x' was reported to need a MTU of 1496.

```
DEV_MTU_1=''              # Adjust MTU size of NIC on VDSL-Modem
                           # Example: DEV_MTU_1='eth1 1496'
```

The config files base.txt and dns_dhcp.txt have to be changed as described in the next chapter.

Configuration Of An Additional NIC For IPTV

In base.txt and dns_dhcp.txt the configuration has to be changed for VLAN and for the second NIC.

Insert the second NIC for IPTV:

```
NET_DRV_N='2'
NET_DRV_1='via-rhine'      # 1. NIC interface for LAN
NET_DRV_2='3c59x'         # 2. NIC - here 3Com for IPTV SetTopBox
```

Now the address range for the second NIC has to be set. We will use 192.168.2.0/24 for the LAN and 192.168.3.0/24 for the second NIC. Entries for the virtual NICs eth1.7 and eth1.8 are needed in addition:

```
IP_NET_N='4'
IP_NET_1='192.168.2.1/24'      # home/office LAN
IP_NET_1_DEV='eth0'
IP_NET_2='192.168.3.1/24'      # iptv LAN
IP_NET_2_DEV='eth2'
IP_NET_3='dhcp'               # dhcp client - IP via dhclient
IP_NET_3_DEV='eth1.8'
IP_NET_3_MAC='00:40:63:da:cf:32' # new MAC (not the MAC of eth1)
IP_NET_4='dhcp'               # eth1.7 connecting to the modem
IP_NET_4_DEV='eth1.7'
IP_NET_4_MAC='00:40:63:da:cf:33' # new MAC (not the MAC of eth1)
```

It is important to change the MAC addresses for eth1.7 and eth1.8 to be different from eth1's one, otherwise - depending on the VDSL net disturbances can occur after forced disconnection.

For the new NIC Internet access must be possible, just as for the first NIC. These additional settings are necessary:

```
PF_INPUT_1='IP_NET_1 ACCEPT'
PF_INPUT_2='IP_NET_2 ACCEPT'
PF_INPUT_3='any 224.0.0.0/4 ACCEPT'
[...]
PF_FORWARD_3='any 224.0.0.0/4 ACCEPT'
PF_FORWARD_5='IP_NET_1 ACCEPT'
PF_FORWARD_6='IP_NET_2 ACCEPT'
[...]
PF_POSTROUTING_1='IP_NET_1 MASQUERADE'
PF_POSTROUTING_2='IP_NET_2 MASQUERADE'
```

For working dynamic DHCP addressing at the new IPTV NIC and to access the set-top box by its name, the following settings in dns_dhcp.txt are required:

```
HOST_10_NAME='igmp'
HOST_10_IP4='192.168.3.1'
HOST_11_NAME='iptv'
HOST_11_IP4='192.168.3.4'
HOST_11_MAC='00:D0:E0:93:49:34'          # MAC Adr T-Home X300T
[...]
DHCP_RANGE_2_NET='IP_NET_2'
DNSDHCP_RANGE_2_START='192.168.3.10'
DNSDHCP_RANGE_2_END='192.168.3.20'
DNSDHCP_RANGE_2_DNS_SERVER1=''
DNSDHCP_RANGE_2_DNS_SERVER2=''
DNSDHCP_RANGE_2_NTP_SERVER=''
DNSDHCP_RANGE_2_GATEWAY=''
```

After configuring the new NIC it is suggested to connect it to a PC to see if Internet access is possible over it. In case of success the new second NIC should be configured correctly.

IGMP Functions

When booting the fli4l router the parameters of the config file proxy.txt are written to the file /etc/igmpproxy.conf, which is read when starting igmpproxy.

In contrast to earlier versions of opt_igmp the IGMP proxy is started at boot and then runs as long as a physical Internet connection is available. The IGMP proxy is not affected by a forced disconnection after 24 hour or manual connect or disconnect of Internet traffic.

IGMP Configuration

OPT_IGMPPROXY 'yes' activates the IGMP proxy package while 'no' deactivates it completely.

IGMPPROXY_DEBUG By specifying 'yes' here messages of the IGMP proxy are sent to syslog.

IGMPPROXY_DEBUG2 By specifying 'yes' here the log level of the IGMP proxy may be increased.

IGMPPROXY_QUICKLEAVE_ON With Quickleave the load in the upstream link can be lowered. If the parameter is enabled by 'yes', this will cause that the Multicast is canceled faster after a channel change and so the downstream load is lowered by the IGMP proxy behaving like a receiver.

If two STBs exist and show the same channel, it may happen (with Quickleave enabled) that the program is interrupted on one box if switched on the second. When using only one STB Quickleave may safely be enabled.

```
IGMPPROXY_QUICKLEAVE_ON='yes'          # activate Quickleave mode
                                         # yes or no; Default: yes
```

IGMPPROXY_UPLOAD_DEV For IPTV operation IGMP proxy requires an upstream and a downstream interface. The upstream interface is the interface of the NIC attached to the VDSL modem. This usually should always remain the same.

With the transfer of IPTV to ID8 eth1.8 instead of ppp0 has to be entered in the configuration file.

```
IGMPPROXY_UPLOAD_DEV='eth1.8'      # Upstream interface; Default: ppp0
                                     # eth1.8 for T-Home/VDSL with id7/id8
```

IGMPPROXY_DOWNLOAD_DEV The interface of the downstream (NIC for IPTV set-top box) is set here dependent on the hardware configuration. For fli4l with second NIC eth2 is the interface for the set-top box.

```
IGMPPROXY_DOWNLOAD_DEV='eth2'      # Downstream interface
```

IGMPPROXY_ALT_N This parameter specifies the number of address ranges for Multicast streams.

IGMPPROXY_ALT_x_NET By the parameter IGMPPROXY_ALT_NET address ranges for Multicast traffic originating outside of the LAN are defined as well as the local address range that connects to the STB.

```
IGMPPROXY_ALT_N='3'                # Number of Multicast sources
IGMPPROXY_ALT_1_NET='239.35.0.0/16' # IPTV streams - always needed
IGMPPROXY_ALT_2_NET='217.0.119.0/24' # needed for T-Home
IGMPPROXY_ALT_3_NET='193.158.34.0/23' # needed for T-Home
                                     # before May 2013 '193.158.35.0/24'
# IGMPPROXY_ALT_4_NET='192.168.3.0/24' # Address range IPTV SetTop-Box/not
                                     # needed anymore
```

IGMPPROXY_WLIST_N With this parameter the number of whitelists for IGMP reports is determined.

IGMPPROXY_WLIST_x_NET :

Using IGMPv3 all addresses may be summarized in one report which in turn will be ignored completely. This leads to a complete shutdown of all Multicast traffic by the IGMP Querier assuming that it is not needed anymore. To avoid this, configuration of whitelists is used. Only Multicast groups in this list will be requested on the WAN side.

```
IGMPPROXY_WLIST_N='1'              # Number of Multicast sources
IGMPPROXY_WLIST_1_NET='239.35.0.0/16' # IPTV streams - always needed
                                     # see above
```

Changes In Other Config Files

As of revision 32955 it is not necessary to adapt the firewall rules for IGMP Proxy and Multicast streams if the standard rules are activated in base.txt (PF_INPUT_ACCEPT_DEF='yes' and PF_FORWARD_ACCEPT_DEF='yes'). The start script will automatically add these rules if OPT_IGMPPROXY='yes' is set.

There will be two rules added to the INPUT chain to let the IGMP Proxy receive incoming messages:

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target  prot opt in  out  source      destination
[...]
0      0 ACCEPT  all  --  *   *    0.0.0.0/0    224.0.0.1    \
/* automatically added for IGMP Proxy */

0      0 ACCEPT  all  --  *   *    0.0.0.0/0    224.0.0.22   \
/* automatically added for IGMP Proxy */
[...]
```

Another rule will be added to the FORWARD chain that enables forwarding of incoming Multicast streams to the media receiver.

```
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target  prot opt in  out  source      destination
[...]
0      0 ACCEPT  all  --  *   *    0.0.0.0/0    239.35.0.0/16 \
/* automatically added for IPTV streams */
[...]
```

If the standard rules are not activated at least the following rules have to be added:

```
PF_INPUT_x='any 224.0.0.1/32 ACCEPT'
PF_INPUT_x='any 224.0.0.22/32 ACCEPT'
[...]
PF_FORWARD_x='any 239.35.0.0/16 ACCEPT'
```

Hint: Despite to earlier versions of the documentation the rules were restricted to the nets really needed. If IPTV does not work as expected feel free to provide additional information concerning the nets used.

Important! By the end of May 2013 the Telekom introduced new classless routes for Entertain (<http://www.onlinekosten.de/forum/showthread.php?t=116415&page=38>). This seems to be caused by the use of more than 256 stations resp. addresses. The DHCP server now transfers routes not contained in the subnet used before. As long as the Telekom does not change its iptv-server-subnet (193.158.34.0/23) a static route may be defined for the vlan8 interface to adapt these changes, otherwise Multicast would not work anymore.

Solution: specify an additional route in base.txt.

```
IP_ROUTE_N='1'
IP_ROUTE_1='193.158.34.0/23 eth1.8'
```

1.1.7 OPT_STUNNEL - Tunneling Connections Over SSL/TLS

The program “stunnel” allows to encapsulate connections otherwise unencrypted in an encrypted SSL/TLS tunnel. This allows safe data exchange over otherwise insecure cleartext protocols. Due to the possibilities of the SSL/TLS protocol, various forms of Client/server validation are possible.

Configuration

OPT_STUNNEL This variable activates support for SSL/TLS tunnels.

Default setting: `OPT_STUNNEL='no'`

Example: `OPT_STUNNEL='yes'`

STUNNEL_DEBUG This variable can be set to configure the logging settings for “stunnel”. Available settings are “yes” (everything is logged), “no” (warnings and errors are logged) or a value between zero and seven indicating the severity of messages with zero for highest and seven for lowest severity. The setting “yes” corresponds to severity seven, while “no” corresponds to severity four.

Default setting: `STUNNEL_DEBUG='no'`

Example 1: `STUNNEL_DEBUG='yes'`

Example 2: `STUNNEL_DEBUG='5'`

STUNNEL_N This variable configures the number of tunnel instances. Each tunnel instance “listens” on a network port “A” and connects to another network port “B” when a connection is established (may as well be on a different machine), then forwards all traffic from “A” to “B”. Whether the data, that arrives at “A” encrypted via SSL/TLS will be decrypted by “stunnel” before forwarding unencrypted to “B” or vice versa is decided by the variable setting in [STUNNEL_x_CLIENT](#) (Page 15).

Default setting: `STUNNEL_N='0'`

Example: `STUNNEL_N='2'`

STUNNEL_x_NAME The name of each tunnel. Must be unique for all configured tunnels.

Example: `STUNNEL_1_NAME='imond'`

STUNNEL_x_CLIENT This variable configures which parts of the communication are encrypted via SSL/TLS. There are two options:

- *Client mode:* The tunnel expects unencrypted data from outside and sends it encrypted to the other end of the tunnel. This corresponds to the setting `STUNNEL_x_CLIENT='yes'`.
- *Server mode:* The tunnel expects data encrypted via SSL/TLS from outside and will send it decrypted to the other end of the tunnel. This is equivalent to setting `STUNNEL_x_CLIENT='no'`.

Tunnels in client mode hence are particularly suitable for connections “to the outside”, i.e. to the (unprotected) Internet because data is encrypted before leaving the local network. Of course the remote site must offer a server that expects data encrypted via SSL/TLS. For example an e-mail client in the LAN only supporting unencrypted POP3 can “talk” to a POP3 over SSL service on the Internet ¹

Tunnels in server mode in reverse are for connections that come “from the outside”, i.e. from the (unprotected) Internet providing encrypted data. If the actual service on the server side is not capable to understand SSL/TLS the data must be decrypted

¹see <http://en.wikipedia.org/wiki/POP3S>

previously. For example the access to the fli4l web GUI can be accomplished via HTTP (HTTPS) encrypted via SSL/TLS by configuring a tunnel on the fli4l receiving HTTP traffic encrypted via SSL/TLS on port 443, then decrypting the data and forwarding it to the local web server `mini_httpd` listening on port 80.

Configurations for these use cases are presented later.

Example: `STUNNEL_1_CLIENT='yes'`

STUNNEL_x_ACCEPT This determines on which port (and address) the tunnel is “listening” for incoming connections. In principle two possibilities exist:

- The tunnel should listen on *all* addresses (on all interfaces). Use the setting “any” in this case.
- The tunnel should only listen to defined addresses. Set this with a reference corresponding to the IP-subnet configured, for example `IP_NET_1_IPADDR` (for IPv4) or `IPV6_NET_2_IPADDR` (for IPv6).

At the end of the address part the port *must* be added, separated by a colon (“:”).

Example 1: `STUNNEL_1_ACCEPT='any:443'`

Example 2: `STUNNEL_1_ACCEPT='IP_NET_1_IPADDR:443'`

Example 3: `STUNNEL_1_ACCEPT='IPV6_NET_2_IPADDR:443'`

Please note that using `IP_NET_x_IPADDR` resp. `IPV6_NET_x_IPADDR` determines the Layer-3-Protocol (IPv4 or IPv6), the choice here *must* match with the settings in the variables `STUNNEL_x_ACCEPT_IPV4` and `STUNNEL_x_ACCEPT_IPV6`. Hence you may not deactivate IPv6 for the tunnel by using `STUNNEL_1_ACCEPT_IPV6='no'` and then listen on an IPv6 address using `STUNNEL_1_ACCEPT='IPV6_NET_2_IPADDR:443'` or vice versa by using (`STUNNEL_1_ACCEPT_IPV4='no'` and `IP_NET_x_IPADDR`). Furthermore, the meaning of “any” depends on the Layer 3 protocols activated (IPv4 or IPv6): of course, the tunnel only listens on addresses belonging to the Layer-3-Protocols activated via `STUNNEL_x_ACCEPT_IPV4` and `STUNNEL_x_ACCEPT_IPV6`.

STUNNEL_x_ACCEPT_IPV4 This variable controls if the IPv4 protocol is used for *incoming* connections to the tunnel. Typically this is the case and this variable should be set to “yes” while “no” ensures that the tunnel only accepts incoming IPv6 connections. However, this requires a valid IPv6 configuration (refer to the documentation for the `ipv6` package for more information).

Default setting: `STUNNEL_x_ACCEPT_IPV4='yes'`

Example: `STUNNEL_1_ACCEPT_IPV4='no'`

STUNNEL_x_ACCEPT_IPV6 Like in `STUNNEL_x_ACCEPT_IPV4` this variable controls whether the IPv6 protocol is used for incoming connections to the tunnel. Typically this is the case if you use the the general IPv6 protocol by using `OPT_IPV6='yes'`. Setting “no” here ensures that the tunnel only accepts incoming IPv4 connections.

Default setting: `STUNNEL_x_ACCEPT_IPV6=<Values from OPT_IPV6>`

Example: `STUNNEL_1_ACCEPT_IPV6='no'`

STUNNEL_x_CONNECT Sets the target of the SSL/TLS tunnel. There are basically three possibilities and all must have the port appended, separated by a colon (“:”):

- A numeric IPv4- or IPv6 address

Example 1: `STUNNEL_1_CONNECT='192.0.2.2:443'`

- The DNS name of an internal host

Example 2: `STUNNEL_1_CONNECT='@webserver:443'`

- The DNS name of an external host

Example 3: `STUNNEL_1_CONNECT='@www.example.com:443'`

If an internal host is entered with both IPv4 and IPv6 address, the IPv4 address is preferred. If an external host is entered with both IPv4 and IPv6 address, then the Layer 3 protocol used depends on which address is first returned by the DNS resolver.

STUNNEL_x_OUTGOING_IP With this optional variable, the *local* address for the *outgoing* connection of the tunnel can be set. This is only useful if the target of the tunnel can be reached over multiple interfaces (routes), i.e. if two concurrent Internet connections are used. Normally, this variable must not be set.

Example: `STUNNEL_1_OUTGOING_IP='IP_NET_1_IPADDR'`

STUNNEL_x_DELAY_DNS If this optional variable is set to “yes”, an external DNS name used in `STUNNEL_x_CONNECT` will not be converted to an address until the *outbound* tunnel is established, meaning the point when the first client has connected locally with the incoming side of the tunnel. This is useful if the target of the tunnel is a computer that can only be reached through a dynamic DNS name and the address behind the name changes frequently, or if an active dialin when starting “stunnel” should be prevented.

Default setting: `STUNNEL_x_DELAY_DNS='no'`

Example: `STUNNEL_1_DELAY_DNS='yes'`

STUNNEL_x_CERT_FILE This variable contains the file name of the certificate for the tunnel to be used. For server mode tunnels (`STUNNEL_x_CLIENT='no'`) this is the server certificate that the client validates against a “Certificate Authority” (CA) if necessary. For client mode tunnels (`STUNNEL_x_CLIENT='yes'`) this is a (usually optional) client certificate that is validated by the server against a CA if necessary.

The certificate must be provided in the so-called PEM format and must be saved below `<config-directory>/etc/ssl/stunnel/`. Only the file name must be stored in this variable, not the path.

For a server mode tunnel the certificate is mandatory!

Example: `STUNNEL_1_CERT_FILE='myserver.crt'`

STUNNEL_x_CERT_CA_FILE This variable contains the file name of the CA certificate to be used for the validation of the certificate of the remote station. Typically clients validate the server’s certificate, vice versa however, is also possible. For details on the validation please refer to the description of the variable [STUNNEL_x_CERT_VERIFY](#) (Page 18).

The certificate must be provided in the so-called PEM format and must be saved below `<config-directory>/etc/ssl/stunnel/`. Only the file name must be stored in this variable, not the path.

Example: `STUNNEL_1_CERT_CA_FILE='myca.crt'`

STUNNEL_x_CERT_VERIFY This variable controls the validation of the certificate of the remote station. There are five options possible:

- *none*: The certificate of the remote station is not validated at all. In this case the variable `STUNNEL_x_CERT_CA_FILE` is empty.
- *optional*: If the remote station provides a certificate it is checked against the CA certificate configured using the variable `STUNNEL_x_CERT_CA_FILE`. If the remote station does *not* provide a certificate this is not an error and the connection is still accepted. This setting is only useful for server mode tunnel because the client tunnel *always* obtain a certificate from the server.
- *onlyca*: The certificate of the remote station is validated against the CA certificate configured in the variable `STUNNEL_x_CERT_CA_FILE`. If the remote station does not provide a certificate or it does not match the configured CA, the connection is rejected. This is useful when a private CA is used, as then all potential peers are known.
- *onlycert*: The certificate of the remote station is compared with the certificate configured in the variable `STUNNEL_x_CERT_CA_FILE`. It is *not* checked against a CA certificate, but it will be ensured that the remote station sends *exactly* the matching (server or client) certificate. The file referenced with the help of the variable `STUNNEL_x_CERT_CA_FILE` in this case does not contain a CA certificate, but a host certificate. This setting ensures that really only a fixed and known peer may connect (server tunnel) or a connection to only a known peer (client tunnel) is established. This is useful for peer-to-peer connections between hosts both under your control, but for which no own CA is used.
- *both*: The certificate of the remote station is compared with the certificate configured by the help of the variable `STUNNEL_x_CERT_CA_FILE` *and* it is also ensured that it matches a CA certificate. The file referenced by the help of the variable `STUNNEL_x_CERT_CA_FILE` in this case contains *both* a CA *and* a host certificate. It is therefore a combination of the settings *onlycert* and *onlyca*. In comparison to the setting *onlycert* connections with expired CA certificate will be rejected (even if the certificate of the peer matches).

Default setting: `STUNNEL_x_CERT_VERIFY='none'`

Example: `STUNNEL_1_CERT_VERIFY='onlyca'`

Use Case 1: Accessing the fli4l-WebGUI via SSL/TLS

This example enhances the fli4l-WebGUI with SSL/TLS access.

```
OPT_STUNNEL='yes'
STUNNEL_N='1'
```

```
STUNNEL_1_NAME='http'
STUNNEL_1_CLIENT='no'
STUNNEL_1_ACCEPT='any:443'
STUNNEL_1_ACCEPT_IPV4='yes'
STUNNEL_1_ACCEPT_IPV6='yes'
STUNNEL_1_CONNECT='127.0.0.1:80'
STUNNEL_1_CERT_FILE='server.pem'
STUNNEL_1_CERT_CA_FILE='ca.pem'
STUNNEL_1_CERT_VERIFY='none'
```

Use Case 2: Controlling two remote fli4l routers via imonc secured by SSL/TLS

The known weaknesses of the imonc/imond protocol for WAN connections (sending passwords in clear text) are bypassed with this example. (The LAN connection to the tunnel of course is vulnerable!)

Configuration of the local fli4l in LAN (client tunnel):

```
OPT_STUNNEL='yes'
STUNNEL_N='2'

STUNNEL_1_NAME='remote-imond1'
STUNNEL_1_CLIENT='yes'
STUNNEL_1_ACCEPT='any:50000'
STUNNEL_1_ACCEPT_IPV4='yes'
STUNNEL_1_ACCEPT_IPV6='yes'
STUNNEL_1_CONNECT='@remote1:50000'
STUNNEL_1_CERT_FILE='client.pem'
STUNNEL_1_CERT_CA_FILE='ca+server1.pem'
STUNNEL_1_CERT_VERIFY='both'

STUNNEL_2_NAME='remote-imond2'
STUNNEL_2_CLIENT='yes'
STUNNEL_2_ACCEPT='any:50001'
STUNNEL_2_ACCEPT_IPV4='yes'
STUNNEL_2_ACCEPT_IPV6='yes'
STUNNEL_2_CONNECT='@remote2:50000'
STUNNEL_2_CERT_FILE='client.pem'
STUNNEL_2_CERT_CA_FILE='ca+server2.pem'
STUNNEL_2_CERT_VERIFY='both'
```

Configuration of the first remote fli4l (server tunnel):

```
OPT_STUNNEL='yes'
STUNNEL_N='1'

STUNNEL_1_NAME='remote-imond'
STUNNEL_1_CLIENT='no'
STUNNEL_1_ACCEPT='any:50000'
STUNNEL_1_ACCEPT_IPV4='yes'
STUNNEL_1_ACCEPT_IPV6='yes'
STUNNEL_1_CONNECT='127.0.0.1:5000'
STUNNEL_1_CERT_FILE='server1.pem'
```

```
STUNNEL_1_CERT_CA_FILE='ca+client.pem'  
STUNNEL_1_CERT_VERIFY='both'
```

Configuration of the second remote fli4l (server tunnel):

```
OPT_STUNNEL='yes'  
STUNNEL_N='1'  
  
STUNNEL_1_NAME='remote-imond'  
STUNNEL_1_CLIENT='no'  
STUNNEL_1_ACCEPT='any:50000'  
STUNNEL_1_ACCEPT_IPV4='yes'  
STUNNEL_1_ACCEPT_IPV6='yes'  
STUNNEL_1_CONNECT='127.0.0.1:5000'  
STUNNEL_1_CERT_FILE='server2.pem'  
STUNNEL_1_CERT_CA_FILE='ca+client.pem'  
STUNNEL_1_CERT_VERIFY='both'
```

A connection to the remote “imond” is established by initiating a connection to the local fli4l on port 50000 (first remote fli4l) resp. 50001 (second remote fli4l). This fli4l then connects via SSL/TLS-Tunnel to each of the remote fli4l’s which in turn forward their data over a third (host internal) connection to the remote “imond” in the end. The settings of the validation ensure that each fli4l only accepts the other fli4l as the connecting counterpart.

List of Figures

1.1	fii4l in a standard configuration	9
1.2	fii4l in an IPTV configuration	9

List of Tables

Index

IGMPPROXY_ALT_N, [13](#)
IGMPPROXY_ALT_x_NET, [13](#)
IGMPPROXY_DEBUG, [12](#)
IGMPPROXY_DEBUG2, [12](#)
IGMPPROXY_DOWNLOAD_DEV, [13](#)
IGMPPROXY_QUICKLEAVE_ON, [12](#)
IGMPPROXY_UPLOAD_DEV, [12](#)
IGMPPROXY_WLIST_N, [13](#)
IGMPPROXY_WLIST_x_NET, [13](#)

OPT_IGMPPROXY, [8](#), [12](#)
OPT_PRIVOXY, [3](#)
OPT_SIPROXD, [7](#)
OPT_SS5, [6](#)
OPT_STUNNEL, [14](#), [15](#)
OPT_TOR, [5](#)
OPT_TRANSPROXY, [7](#)

PRIVOXY_MENU, [3](#)
PRIVOXY_N, [3](#)
PRIVOXY_x_ACTIONDIR, [3](#)
PRIVOXY_x_ALLOW_N, [3](#)
PRIVOXY_x_ALLOW_x, [3](#)
PRIVOXY_x_CONFIG, [4](#)
PRIVOXY_x_HTTP_PROXY, [4](#)
PRIVOXY_x_LISTEN, [3](#)
PRIVOXY_x_LOGDIR, [4](#)
PRIVOXY_x_LOGLEVEL, [4](#)
PRIVOXY_x SOCKS_PROXY, [4](#)
PRIVOXY_x_TOGGLE, [4](#)

SS5_ALLOW_N, [6](#)
SS5_ALLOW_x, [7](#)
SS5_LISTEN_N, [6](#)
SS5_LISTEN_x, [6](#)
STUNNEL_DEBUG, [15](#)
STUNNEL_N, [15](#)
STUNNEL_x_ACCEPT, [16](#)
STUNNEL_x_ACCEPT_IPV4, [16](#)
STUNNEL_x_ACCEPT_IPV6, [16](#)
STUNNEL_x_CERT_CA_FILE, [17](#)
STUNNEL_x_CERT_FILE, [17](#)
STUNNEL_x_CERT_VERIFY, [18](#)
STUNNEL_x_CLIENT, [15](#)
STUNNEL_x_CONNECT, [16](#)
STUNNEL_x_DELAY_DNS, [17](#)
STUNNEL_x_NAME, [15](#)
STUNNEL_x_OUTGOING_IP, [17](#)

TOR_ALLOW_N, [5](#)
TOR_ALLOW_x, [5](#)
TOR_CONTROL_PASSWORD, [6](#)
TOR_CONTROL_PORT, [6](#)
TOR_DATA_DIR, [6](#)
TOR_HTTP_PROXY, [6](#)
TOR_HTTP_PROXY_AUTH, [6](#)
TOR_HTTPS_PROXY, [6](#)
TOR_HTTPS_PROXY_AUTH, [6](#)
TOR_LISTEN_N, [5](#)
TOR_LISTEN_x, [5](#)
TOR_LOGFILE, [6](#)
TOR_LOGLEVEL, [6](#)
TRANSPROXY_ALLOW_N, [7](#)
TRANSPROXY_ALLOW_x, [7](#)
TRANSPROXY_LISTEN_N, [7](#)
TRANSPROXY_LISTEN_x, [7](#)
TRANSPROXY_TARGET_IP, [7](#)
TRANSPROXY_TARGET_PORT, [7](#)