fli4l – flexible internet router for linux
Version 3.10.5

The fli4l-Team
email: team@fli4l.de

February 16, 2016

Contents

1. Documentation of the base package

1.1. Introduction

fli4l is a Linux-based router, capable of handling ISDN, DSL, UMTS, and ethernet connections, with little hardware requirements: an USB stick used for booting, an Intel Pentium MMX processor, 64 MiB RAM as well as (at least) one ethernet network adapter are completely sufficient. The necessary boot medium can be created under Linux, Mac OS X or MS Windows. You don't need any specific Linux knowledge, but it is definitely helpful. However, you should possess basic knowledge about networking, TCP/IP, DNS, and routing. For developing your own extensions exceeding the basic configuration, you will need a working Linux system as well as Linux skills.

fli4l supports various boot media, among them USB sticks, hard disks, CDs, and last but not least booting over the network. An USB stick is in many respects ideal: Today, almost every PC can boot from it, it is relatively cheap, it is big enough, and installing fli4l onto it is relatively easy under both MS Windows and Linux. In contrast to a CD it is writable and thus additionally able to hold non-volatile configuration data (as e.g. DHCP leases).

- General features
  - Creation of boot media under Linux (Page 249), Mac OS X (Page 249), and MS Windows (Page 252)
  - Configuration through flat ASCII/UTF-8 files
  - Support for IP masquerading and port forwarding
  - Least Cost Routing (LCR): automatic provider selection based on daytime
  - Displaying/Computing/Logging of connection times and costs
  - MS Windows/Linux client imonc talking to imond and telmond
  - Upload of updated configuration files via MS Windows client imonc or via SCP under Linux
  - Boot media use the VFAT file system as permanent storage
  - Packet filter: External access to blocked ports is logged
  - Uniform mapping of WAN interfaces to so-called circuits
  - Running ISDN and DSL/UMTS circuits in parallel is possible
- Router basics
  - Linux kernel 3.14
  - Packet filter and IP masquerading
  - Local DNS server in order to reduce the number of DNS queries to external DNS servers

  - Remotely accessible imond server daemon for monitoring and controlling Least Cost Routing
  - Remotely accessible telmond server daemon logging incoming phone calls
- Ethernet support
  - Up-to-date network device drivers: Support for more than 140 adapter types
- DSL support
  - Roaring Penguin PPPoE driver supporting Dial-on-Demand (can be switched off)
  - PPTP for DSL providers in Austria and the Netherlands
- ISDN support
  - Support for some 60 adapter types
  - Multiple possibilities for ISDN connectons: incoming/outgoing/callback, raw/point-to-point (ppp)
  - Channel bundling: automatic band width adaptation or manual activation of the second channel using MS Windows/Linux client software
- Optional software packages
  - DNS server
  - DHCP server
  - SSH server
  - Simple online/offline display using a LED
  - Serial console
  - Minimalistic Web server for ISDN and DSL monitoring as well as for reconfiguring and/or updating the router
  - Ability to let external hosts access LAN hosts in a controlled manner
  - Support for PCMCIA cards (called PC cards nowadays)
  - Logging of system messages
  - Configuration of ISAPnP cards by the use of isapnp tools
  - Additional tools for debugging
  - Configuration of the serial port
  - Rescue system for remote administration over ISDN
  - Software for displaying configurable information on an LCD, e.g. transmission rates, CPU load etc.
  - PPP server/router over the serial port
  - ISDN modem emulator over the serial port
  - Print server
  - Time synchronization with external time servers

  - Execution of user-defined commands on incoming phone calls (e.g. to perform Internet dial-up)
  - Support for IP aliasing (multiple IP addresses per network interface)
  - VPN support
  - IPv6 support
  - WLAN support: fli4l can be an access point as well as a client
  - RRD tool for monitoring the fli4l
  - and much more...

- Hardware requirements
  - Intel Pentium processor with MMX support
  - 64 MiB RAM, better 128 MiB
  - Ethernet network adapter
  - ISDN: supported ISDN adapter
  - an USB stick, an ATA hard disk or a CF card (which is accessed the same way as an ATA hard disk); alternatively, booting from a CD is also possible

- Software requirements

  The following tools are required on Linux systems:
  - GCC and GNU make
  - syslinux
  - mtools (mcopy)

  No additional tools are required on MS Windows systems, all necessary tools are provided by fli4l.

Last but not least, the client utility imonc exists for controlling the router and for displaying the router's state. This tool is available for MS Windows (windows/imonc.exe) and also for Linux (unix/gtk-imonc).

And now ...

Have fun with fli4l!

Frank Meyer and the fli4l team
email: team@fli4l.de

2. Setup and Configuration

2.1. Unpacking the archives

Under Linux:

```
tar xvfz fli4l-3.10.5.tar.gz
```

If this does not work, try the following:

```
gzip -d < fli4l-3.10.5.tar.gz | tar xvf -
```

If you unpack the current version into a directory which already contains fli4l files from a previous installation, you should execute `mkfli4l.sh -c`:

```
 cd fli4l-3.10.5
sh mkfli4l.sh -c
```

However, we recommend to use a fresh directory for a new version as you can easily take over the configuration with a file comparing tool.

If you use a MS Windows system, you can extract the compressed tar archive e.g. with WinZip. You have to check, however, that all files are extracted together *with* their corresponding directories (there is a WinZip setting to achieve this). Also, you have to disable the so-called "Smart TAR CR conversion" under *Settings* ⇒*Configuration*, as some important files are corrupted during extraction otherwise.

Alternatively, we recommend using the OpenSource application 7-Zip (http://www.7-zip.org/) which is as powerful as WinZip.

The following files are installed in the subdirectory `fli4l-3.10.5/`:

- Documentation:
  - doc/deutsch/* German documentation
  - doc/english/* English documentation
  - doc/french/* French documentation
- Configuration:
  - config/*.txt Configuration files, these ones have to be edited to suit your needs
- Scripts/Procedures:
  - mkfli4l.sh Creates boot media or files: Linux/Unix version
  - mkfli4l.bat Creates boot media: Windows version
- Kernel/Boot files:

  - img/kernel Linux kernel
  - img/boot*.msg bootscreen texts

- Additional packages:
  - opt/*.txt These ones describe which files will be included in the opt.img archive.
  - opt/etc/* Default configuration files for many applications (normally, these files need not be edited)
  - opt/files/* Optional kernel modules, files, and programs

- Source code:
  - src/* Source code/tools for Linux, see src/README

- Programs:
  - unix/mkfli4l* Creates boot medium: Linux/Unix version
  - windows/* Creates boot medium: Windows version
  - unix/imonc* imond client for Unix/Linux
  - windows/imonc/* imond client for Windows

2.2. Configuration

2.2.1. Editing the configuration files

To configurate fli4l, you only have to adapt the files config/*.txt. We recommend to make a copy of the "config" directory and perform the adaption within this copy. So you will be able to compare your own configuration with the distributed one later and you are also able to manage multiple configurations. Comparing two configurations is relatively simple by using an adequate tool (i.e. "diff" under *nix). Let's assume that your own config files reside in a directory named my_config under the fli4l directory. In this case you could execute:

```
~/src/fli4l> diff -u {config,my_config}/build/full_rc.cfg | grep '^[+-]'
--- config/build/full_rc.cfg    2014-02-18 15:34:39.085103706 +0100
+++ my_config/build/full_rc.cfg        2014-02-18 15:34:31.094317441 +0100
-PASSWORD='/P6h4iOIN5Bbc'
+PASSWORD='3P8F3KbjYgzUc'
-NET_DRV_1='ne2k-pci'
+NET_DRV_1='pcnet32'
-START_IMOND='no'
+START_IMOND='yes'
-OPT_PPPOE='no'
+OPT_PPPOE='yes'
-PPPOE_USER='anonymous'
-PPPOE_PASS='surfer'
+PPPOE_USER='me'
+PPPOE_PASS='my-passwd'
-OPT_SSHD='no'
+OPT_SSHD='yes'
```

You can also see by this example that a simple DSL router can be configured without much effort, even if you feel at first overwhelmed by the sheer amount of possible settings.

2.2.2. Configuration via a special configuration file

Due to the module concept of fli4l, the configuration is distributed across different files. As editing these separate files may become tedious, it is possible to store the configuration into a single file called *<config directory>/_fli4l.txt*. This file is read in addition to the other configuration files and its contents override any settings found in the other configuration files. Recall the example above: In order to configure a simple DSL router, we could simply write the following lines into this file:

```
PASSWORD='3P8F3KbjYgzUc'
NET_DRV_N='1'
NET_DRV_1='pcnet32'
START_IMOND='yes'
OPT_PPPOE='yes'
PPPOE_USER='me'
PPPOE_PASS='my-passwd'
OPT_SSHD='yes'
```

You should avoid to mix both flavours of configuration.

2.2.3. Variables

You will notice that teh lines of some variables are prefixed with a '#'. and thus are commented. If this is the case a reasonable default setting is already in effect. Those defaults are documented for each variable. If you wish to set another value delete the '#' at the beginning of the line and put your value between the apostrophs.

2.3. Setup flavours

Previous versions of fli4l only supported booting from a floppy disk which is not possible anymore due to causes already described. There are many alternative possible nowadays, amongst them using an USB stick.

Many other boot media (CD, HD, network, Compact-Flash, DoC, ...) exist and fli4l may also be installed permanently on some of them (obviously only the read-write ones). fli4l may be booted in three different ways:

Single Image The boot loader loads the linux kernel and then fli4l in a single image. After that, fli4l is able to continue the boot process without the need to access other boot media. Examples are the boot types *integrated*, *attached*, *netboot*, and *cd*.

Split Image The boot loader loads the linux kernel and then a rudimental fli4l image which mounts the boot medium in a first step, then loads the configuration and the remaining files from an archive residing on that mounted medium. Examples for this are the boot types *hd (Type A)*, *ls120*, *attached*, and *cd-emul*.

Installation on a Medium The boot loader loads the linux kernel and then a rudimental fli4l image which mounts an existing fli4l installation without the need to extract any further archives. An example for this is a type B hard disk installation.

Before you try the more advanced installation procedures you should make yourself comfortable with fli4l by setting up a minimal version. If you want to use your fli4l as an answering machine or a HTTP-proxy later on, you already feel confident and have the experience of setting up a basic running system.

That said, four variants of installations are possible:

USB-Stick Router on an USB stick

CD-router Router on a CD

Netzwerk Network boot

HD-Installation Typ A Router on a hard disk, CF, DoC – only one FAT-Partition

HD-Installation Typ B Router on a hard disk, CF, DoC – one FAT- and one ext3-Partition

2.3.1. Router on a USB-Stick

USB-Sticks are addressed as harddisks by Linux hence the same explanations as for the harddisk installation are valid here. Please note that the according drivers for the USB port have to be loaded via `OPT_USB` in order to access the stick with `OPT_HDINSTALL`.

2.3.2. Router on a CD, or network boot

All necessary files are on the boot medium and are extracted to a dynamically sized RAM disk while booting. Using a minimalistic configuration, it is possible to run the router with only 64 MiB of RAM. The maximum setup is only limited by the capacity of the boot medium and available RAM.

2.3.3. Type A: Router on hard disk—only one FAT partition

This corresponds to the CD version, with the only difference of the files residing on a hard disk instead, the term “hard disk” also enclosing Compact Flash from 8 MiB upwards and other devices which are accessed like hard disks under Linux. As of fli4l 2.1.4, you can also use DiskOnChip Flash memory from M-Sys or SCSI hard disks.

The limit for the archive opt.img is removed by disk capacity, but all these files have to be installed into a RAM disk of suitable size during the boot process. This increases the necessary amount of RAM if you use many software packages.

In order to update software packages (i.e. the archive opt.img and the configuration rc.cfg over the network), the FAT partition has to provide enough space for the kernel, the RootFS and TWICE the size of the opt.img archive! If you also want to enable the recovery option, the required space is increased one more time by the size of the opt.img archive.

2.3.4. Type B: Router on hard disk—one FAT and one ext3 partition

In contrast to type A, most of the files are not put into the RAM disk. Instead, they are copied from the opt.img archive to the ext3 partition on the hard disk at the very first start after the initial installation or an update. On successive reboots they are loaded from the ext3 partition. Using this type of installation, the amount of RAM needed for running the router

is the smallest, such that running the router with very low memory is possible in the majority of cases.

You can find further information on the hard disk installation in the documentation of the HD package (a separate download) starting at the description of the configuration variable `OPT_HDINSTALL`.

3. Base configuration

Since fli4l 2.0 the distribution is designed to be modular and consists of multiple packages which have to be downloaded separately. The package `fli4l-3.10.5.tar.gz` contains only the base software for a pure ethernet router. For DSL, ISDN, and other software you will have to download further packages and extract them into the directory `fli4l-3.10.5/`. In order to allow free choice of the fli4l's linux system kernel, the kernel has been removed from the base package and was put into an own package. This implies that at least both the base and kernel packages are required. Table 3.1 gives an overview of additional software packages.

The files necessary for configuring the fli4l router are placed in the directory `config/`. They are described later in this chapter.

These files can be edited with a *simple* text editor, or alternatively with an editor specially designed for fli4l. Miscellaneous editors can be found under

http://www.fli4l.de/en/download/additional-packages/addons/.

If you need to adapt/extend the fli4l system in addition to the possible settings described below, you will need a working Linux system in order to adjust the RootFS. The file `src/README` will provide you with more information.

Table 3.1.: Overview of additional packages

Archive to download	Package
fli4l-3.10.5	BASE, required!
kernel_3_14	Kernel 3.14.z, recommended
kernel_3_14_virt	Kernel 3.14-virt, alternative, for use in virtual environments
kernel_3_14_nonfree	Kernel 3.14-nonfree, alternative, with support for non-GPL-drivers
kernel_3_14_virt_nonfree	Kernel 3.14-virt-nonfree, alternative, combines both above
fli4l-3.10.5-doc	Complete documentation
advanced_networking	Extended network configuration
chrony	Time server/client
dhcp_client	Miscellaneous DHCP clients
dns_dhcp	DNS und DHCP servers
dsl	DSL router (PPPoE, PPTP)
dyndns	Support for DYNDNS services
easycron	Time planning service
hd	Needed for hard disk installation
httpd	Minimalistic Web server
imonc_windows	Windows imonc client
imonc_unix	GTK/Unix imonc client
ipv6	Internet Protocol Version 6
isdn	ISDN router
lcd	LCD driver software + state display
lpdsrv	Print server (without spooler)
openvpn	VPN support
pcmcia	Support for PCMCIA (PC cards)
ppp	PPP connection over serial port
proxy	Proxy server
qos	Quality of Service
sshd	SSH server
tools	Miscellaneous Linux tools
umts	Connection to the Internet via UMTS
usb	USB support
wlan	Support for WLAN cards

3.1. Example file

The content of the example file `base.txt` in directory `config/` is as follows:

```
##-----------------------------------------------------------------------------
## fli4l __FLI4LVER__ - configuration for package "base"
##
##  P L E A S E  R E A D  T H E  D O C U M E N T A T I O N !
##
##  B I T T E  U N B E D I N G T  D I E  D O K U M E N T A T I O N  L E S E N !
```

```
##
##----------------------------------------------------------------------------
## Creation:     26.06.2001  fm
## Last Update:  $Id: base.txt 44415 2016-02-04 06:52:48Z kristov $
##
## Copyright (c) 2001-2016 - Frank Meyer, fli4l-Team <team@fli4l.de>
##
## This program is free software; you can redistribute it and/or modify
## it under the terms of the GNU General Public License as published by
## the Free Software Foundation; either version 2 of the License, or
## (at your option) any later version.
##----------------------------------------------------------------------------

#-----------------------------------------------------------------------------
# General settings:
#-----------------------------------------------------------------------------
HOSTNAME='fli4l'                # name of fli4l router
PASSWORD='fli4l'                # password for root login (console, sshd,
                                # imond)
BOOT_TYPE='hd'                  # boot device: hd, cd, ls120, integrated,
                                # attached, netboot, pxeboot
LIBATA_DMA='disabled'           # Use DMA on ATA Drives ('enabled') or not
                                # ('disabled'). The default 'disabled' allows
                                # ancient IDE CF cards to be booted from.
                                # Use 'enabled' if you boot from a VirtualBox's
                                # virtual device.
MOUNT_BOOT='rw'                 # mount boot device: ro, rw, no
BOOTMENU_TIME='5'               # waiting time of bootmenu in seconds
                                # before activating normal boot
TIME_INFO='MEZ-1MESZ,M3.5.0,M10.5.0/3'
                                # description of local time zone,
                                # don't touch without reading documentation
KERNEL_VERSION='3.14.60'        # kernel version
KERNEL_BOOT_OPTION=''           # append option to kernel command line
COMP_TYPE_OPT='xz'              # compression algorithm if compression is
                                # enabled for OPT archive;
                                # NOTE that some boot types may disallow
                                # some compression algorithms
IP_CONNTRACK_MAX=''             # override maximum limit of connection
                                # tracking entries
POWERMANAGEMENT='acpi'          # select pm interface: none, acpi, apm, apm_rm
                                # apm_rm switches to real mode before invoking
                                # apm power off

#-----------------------------------------------------------------------------
# Localisation
#-----------------------------------------------------------------------------
LOCALE='de'                     # defines the default language for several
                                # components, such as httpd

#-----------------------------------------------------------------------------
# Console settings (serial console, blank time, beep):
#-----------------------------------------------------------------------------
```

```
CONSOLE_BLANK_TIME=''           # time in minutes (1-60) to blank
                                # console; '0' = never, '' = system default
BEEP='yes'                      # enable beep after boot and shutdown
SER_CONSOLE='no'                # use serial interface instead of or as
                                # additional output device and main input
                                # device
SER_CONSOLE_IF='0'              # serial interface to use, 0 for ttyS0 (COM1)
SER_CONSOLE_RATE='9600'         # baudrate for serial console

#-------------------------------------------------------------------------
# Debug Settings:
#-------------------------------------------------------------------------
DEBUG_STARTUP='no'              # write an execution trace of the boot

#-------------------------------------------------------------------------
# Keyboard layout
#-------------------------------------------------------------------------
KEYBOARD_LOCALE='auto'          # auto: use most common keyboard layout for
                                # the language specified in 'LOCALE'
#OPT_MAKEKBL='no'               # set to 'yes' to make a new local keyboard
                                # layout map on the fli4l-router

#-------------------------------------------------------------------------
# Ethernet card drivers:
#-------------------------------------------------------------------------
#
#  please see file base_nic.list in your config-dir or read the documentation
#
#
#  If you need a dummy device, use 'dummy' as your NET_DRV
#  and IP_NET_%_DEV='dummy<number>' as your device
#
#-------------------------------------------------------------------------
NET_DRV_N='1'                   # number of ethernet drivers to load, usually 1
NET_DRV_1='ne2k-pci'            # 1st driver: name (e.g. NE2000 PCI clone)
NET_DRV_1_OPTION=''             # 1st driver: additional option
NET_DRV_2='ne'                  # 2nd driver: name (e.g. NE2000 ISA clone)
NET_DRV_2_OPTION='io=0x320'     # 2nd driver: additional option

#-------------------------------------------------------------------------
# Ether networks used with IP protocol:
#-------------------------------------------------------------------------
IP_NET_N='1'                    # number of IP ethernet networks, usually 1
IP_NET_1='192.168.6.1/24'       # IP address of your n'th ethernet card and
                                # netmask in CIDR (no. of set bits)
IP_NET_1_DEV='eth0'             # required: device name like ethX

#-------------------------------------------------------------------------
# Additional routes, optional
#-------------------------------------------------------------------------
IP_ROUTE_N='0'                  # number of additional routes
IP_ROUTE_1='192.168.7.0/24 192.168.6.99'
                                # network/netmaskbits gateway
```

```
IP_ROUTE_2='0.0.0.0/0 192.168.6.99'
                                # example for default-route

#-------------------------------------------------------------------------
# Packet filter configuration
#-------------------------------------------------------------------------

PF_INPUT_POLICY='REJECT'        # be nice and use reject as policy
PF_INPUT_ACCEPT_DEF='yes'       # use default rule set
PF_INPUT_LOG='no'               # don't log at all
PF_INPUT_LOG_LIMIT='3/minute:5' # log 3 events per minute; allow a burst of 5
                                # events
PF_INPUT_REJ_LIMIT='1/second:5' # reject 1 connection per second; allow a burst
                                # of 5 events; otherwise drop packet
PF_INPUT_UDP_REJ_LIMIT='1/second:5'
                                # reject 1 udp packet per second; allow a burst
                                # of 5 events; otherwise drop packet
PF_INPUT_N='1'                  # number of INPUT rules
PF_INPUT_1='IP_NET_1 ACCEPT'    # allow all hosts in the local network to
                                # access the router
PF_INPUT_2='tmpl:samba DROP NOLOG'
                                # drop (or reject) samba access
PF_INPUT_2_COMMENT='no samba traffic allowed'
                                # without logging, otherwise the log file will
                                # be filled with useless entries

PF_FORWARD_POLICY='REJECT'      # be nice and use reject as policy
PF_FORWARD_ACCEPT_DEF='yes'     # use default rule set
PF_FORWARD_LOG='no'             # don't log at all
PF_FORWARD_LOG_LIMIT='3/minute:5'
                                # log 3 events per minute; allow a burst of 5
                                # events
PF_FORWARD_REJ_LIMIT='1/second:5'
                                # reject 1 connection per second; allow a burst
                                # of 5 events; otherwise drop packet
PF_FORWARD_UDP_REJ_LIMIT='1/second:5'
                                # reject 1 udp packet per second; allow a burst
                                # of 5 events; otherwise drop packet
PF_FORWARD_N='2'                # number of FORWARD rules
PF_FORWARD_1='tmpl:samba DROP'  # drop samba traffic if it tries to leave the
                                # subnet
PF_FORWARD_2='IP_NET_1 ACCEPT'  # accept everything else

PF_OUTPUT_POLICY='ACCEPT'       # default policy for outgoing packets
PF_OUTPUT_ACCEPT_DEF='yes'      # use default rule set
PF_OUTPUT_LOG='no'              # don't log at all
PF_OUTPUT_LOG_LIMIT='3/minute:5'
                                # log 3 events per minute; allow a burst of 5
                                # events
PF_OUTPUT_REJ_LIMIT='1/second:5'
                                # reject 1 connection per second; allow a burst
                                # of 5 events; otherwise drop packet
PF_OUTPUT_UDP_REJ_LIMIT='1/second:5'
```

```
                                # reject 1 udp packet per second; allow a burst
                                # of 5 events; otherwise drop packet
PF_OUTPUT_N='0'                 # number of OUTPUT rules

PF_POSTROUTING_N='1'            # number of POSTROUTING rules
PF_POSTROUTING_1='IP_NET_1 MASQUERADE'
                                # masquerade traffic leaving the subnet

PF_PREROUTING_N='0'             # number of PREROUTING rules
PF_PREROUTING_1='1.2.3.4 dynamic:22 DNAT:@client2'
                                # forward ssh connections coming from 1.2.3.4
                                # to client2

PF_PREROUTING_CT_ACCEPT_DEF='yes'
                                # use default rule set
PF_PREROUTING_CT_N='1'          # number of conntrack PREROUTING rules
PF_PREROUTING_CT_1='tmpl:ftp IP_NET_1 HELPER:ftp'
                                # associate FTP conntrack helper for active FTP
                                # forwarded from within the LAN
PF_PREROUTING_CT_2='tmpl:ftp any dynamic HELPER:ftp'
                                # associate FTP conntrack helper for active FTP
                                # forwarded to the router's external IP

PF_OUTPUT_CT_ACCEPT_DEF='yes'   # use default rule set
PF_OUTPUT_CT_N='0'              # number of conntrack OUTPUT rules
PF_OUTPUT_CT_1='tmpl:ftp HELPER:ftp'
                                # associate FTP conntrack helper for outgoing
                                # active FTP on the router (this rule is added
                                # automatically by the tools package if
                                # OPT_FTP='yes' and FTP_PF_ENABLE_ACTIVE='yes')

PF_USR_CHAIN_N='0'              # number of user-defined rules

#-------------------------------------------------------------------------------
# Domain configuration:
# settings for DNS, DHCP server and HOSTS -> see package DNS_DHCP
#-------------------------------------------------------------------------------
DOMAIN_NAME='lan.fli4l'         # your domain name
DNS_FORWARDERS='194.8.57.8'     # DNS servers of your provider,
                                # e.g. ns.n-ix.net

# optional configuration for the host-entry of the router in /etc/hosts
#HOSTNAME_IP='IP_NET_1_IPADDR'  # IP to bind to HOSTNAME
#HOSTNAME_ALIAS_N='0'           # how many ALIAS names for the router
#HOSTNAME_ALIAS_1='router.lan.fli4l'
                                # first ALIAS name
#HOSTNAME_ALIAS_2='gateway.my.lan'
                                # secound ALIAS name

#-------------------------------------------------------------------------------
# imond configuration:
#-------------------------------------------------------------------------------
START_IMOND='no'                # start imond: yes or no
```

```
IMOND_PORT='5000'               # port (tcp), don't open it to the outside
IMOND_PASS=''                   # imond-password, may be empty
IMOND_ADMIN_PASS=''             # imond-admin-password, may be empty
IMOND_LED=''                    # tty for led: com1 - com4 or empty
IMOND_BEEP='no'                 # beep if connection is going up/down
IMOND_LOG='no'                  # log /var/log/imond.log: yes or no
IMOND_LOGDIR='auto'             # log-directory, e.g. /var/log or auto for
                                # saving in auto-detected savedir
IMOND_ENABLE='yes'              # accept "enable/disable" command
IMOND_DIAL='yes'                # accept "dial/hangup" command
IMOND_ROUTE='yes'               # accept "route" command
IMOND_REBOOT='yes'              # accept "reboot" command

#------------------------------------------------------------------------------
# Generic circuit configuration:
#------------------------------------------------------------------------------
IP_DYN_ADDR='yes'               # use dyn. IP addresses (most providers do)
DIALMODE='auto'                 # standard dialmode: auto, manual, or off

#------------------------------------------------------------------------------
# optional package: syslogd
#------------------------------------------------------------------------------
#OPT_SYSLOGD='no'               # start syslogd: yes or no
#SYSLOGD_RECEIVER='yes'         # receive messages from network
SYSLOGD_DEST_N='1'              # number of destinations
SYSLOGD_DEST_1='*.* /dev/console'
                                # n'th prio & destination of syslog msgs
SYSLOGD_DEST_2='*.* @192.168.6.2'
                                # example: loghost 192.168.6.2
SYSLOGD_DEST_3='kern.info /var/log/dial.log'
                                # example: log infos to file

SYSLOGD_ROTATE='no'             # rotate syslog-files once every day
SYSLOGD_ROTATE_DIR='/data/syslog'
                                # move rotated files to ....
SYSLOGD_ROTATE_MAX='5'          # max number of rotated syslog-files

#------------------------------------------------------------------------------
# Optional package: klogd
#------------------------------------------------------------------------------
#OPT_KLOGD='no'                 # start klogd: yes or no

#------------------------------------------------------------------------------
# Optional package: logip
#------------------------------------------------------------------------------
#OPT_LOGIP='no'                 # logip: yes or no
LOGIP_LOGDIR='auto'             # log-directory, e.g. /boot or auto-detected

#------------------------------------------------------------------------------
# Optional package: y2k correction
#------------------------------------------------------------------------------
#OPT_Y2K='no'                   # y2k correction: yes or no
Y2K_DAYS='0'                    # correct hardware y2k-bug: add x days
```

```
#-------------------------------------------------------------------------
# Optional package: PNP
#-------------------------------------------------------------------------
#OPT_PNP='no'                    # install isapnp tools: yes or no
```

Please note that this file is stored with DOS line endings, i.e. each line contains an additional carriage return (CR) at the end. Since most Unix editors can handle such files it was decided to use this style, as Windows editors typically do have problems if no CR/LF line endings are used!

If your favourite Unix/Linux editor does not like editing some configuration file due to the DOS line endings, you can convert the DOS line endings to Unix ones with the following command before you start editing the file:

```
sh unix/dtou config/base.txt
```

For the creation of the boot media it is irrelevant whether the file contains DOS oder Unix line endings. They are always converted to Unix style when being written to the boot media.

But let's proceed to the contents ...

3.2. General settings

HOSTNAME Default Setting: `HOSTNAME='fli4l'`

At the very beginning you should choose a name for your fli4l router.

PASSWORD Default Setting: `PASSWORD='fli4l'`

This password is needed for logging on to the router—regardless whether you use a keyboard attached to the router or a remote SSH console (for the latter you will need the sshd package). The minimum password length is 1, the maximum 126 characters.

BOOT_TYPE Default setting: `BOOT_TYPE='hd'`

`BOOT_TYPE` determines the boot medium in the broadest sense and affects the drivers (kernel modules) and start scripts being included in the RootFS. A short description of the boot process for better understanding:

- The BIOS of the computer loads and starts the boot loader on the boot medium.
- The boot loader (typically syslinux) extracts, loads, and starts the kernel.
- The kernel extracts the RootFS (= the basic file system containing tools and scripts needed for booting), mounts the RootFS and begins to execute the start scripts.
- Depending on `BOOT_TYPE`, the start scripts loads the kernel modules for the boot medium, mounts the boot partition, and extracts the OPT archive (`opt.img`) containing additional programs.
- Subsequently, fli4l starts to configure the individual services.

The following values are valid for `BOOT_TYPE` at the moment:

ls120 Choose this to boot from LS120/240 and ZIP disks.

hd Choose this to boot from a hard disk. You will find more information in the Documentation (Page 116) of the HD package.

cd Choose this to boot from CD-ROM. With this setting, the ISO image fli4l.iso will be created which you have to burn onto CD with your favourite CD burning application. Please pay attention to choosing the right driver for your CD drive.

integrated Choose this if you do not plan to use a conventional boot medium but e.g. want to boot over a network. This setting integrates the OPT archive into the RootFS so the kernel can extract everything at once and does not need not mount a boot medium.
Note: You cannot perform a remote update of your fli4l router in this case.

attached This setting is similar to **integrated** but it does not integrate the contents of the OPT archive into the RootFS; rather, the OPT archive is put "as is" into the `/boot` directory. From there it will be extracted during the boot process.
Apart from that, the caveats described for `integrated` apply to this boot type as well.

netboot This setting corresponds to **integrated**. However, the script `mknetboot.sh` is additionally run to create an image for booting over the local network. Please read the wiki https://ssl.nettworks.org/wiki/display/f/fli4l+und+Netzboot for further information.

pxeboot Two images are generated, kernel and rootfs.img. These are the two files to be loaded by the PXE bootloader. During execution the local tftp directory may be specified and in addition a subdirectory in the tftp directory (–pxesubdir). Refer to the Wiki here as well: https://ssl.nettworks.org/wiki/display/f/fli4l+und+Netzboot.

Note: How to configure fli4l as a boot-server (pxe/tftp) you can find in the documentation of opt dns_dhcp!

LIBATA_DMA Default Setting: `LIBATA_DMA='disabled'`

This options selects if DMA is used for libata based Devices. It is needed for example for incompletely wired IDE to CompactFlash Adapters. Select 'enabled' to use DMA.

MOUNT_BOOT Default Setting: `MOUNT_BOOT='rw'`

This variable specifies how to mount the boot medium. There are three possibilities:

rw – Read/Write – Writing and reading is possible

ro – Read-Only – Only reading is possible

no – None – Medium will be unmounted after booting and can then be removed if desired

Some configurations require mounting the boot medium read/write, e.g. if you want to run a DHCP server or if you want the imond log file to be stored on the boot medium.

BOOTMENU_TIME Default setting: `BOOTMENU_TIME='20'`

This variable controls how LONG the syslinux boot loader should wait until the default installation is booted automatically.

The `OPT_RECOVER` variable of the HD package allows you to activate a function which enables you to create a recovery installation from a working installation. This recovery installation can be activated in the boot menu by choosing the recovery version.

If this variable contains the value '0', the syslinux boot loader will wait indefinitely until the user chooses either the default or the recovery installation!

TIME_INFO Default Setting: `TIME_INFO='MEZ-1MESZ,M3.5.0,M10.5.0/3'`

Normally, Unix operating systems use the UTC (Coordinated Universal Time) for clocks running under their control, and so does fli4l. The UTC is consistent around the world and has to be converted to local time before use. By using `TIME_INFO`, you provide the necessary information for fli4l about your time zone, its difference to UTC, and about daylight saving time. Your local hardware clock must be set to UTC (corresponds to London Standard Time) in order to make these settinge effective. Alternatively, you may use the chrony Package which allows fli4l to synchronize its clock with an external time server (time servers always provide the current time in UTC).

The meaning of the possible settings `TIME_INFO` are as follows:

```
TIME_INFO='MEZ-1MESZ,M3.5.0,M10.5.0/3'
```

- *MEZ-1:* "MEZ" is the German abbreviation for "Central European Time" (you could also use "CET" here). The "-1" means that *MEZ-1=UTC*, i.e. the MEZ is one hour ahead of UTC.
- *MESZ:* "MESZ" is the abbreviation for "Central European Summer Time" and means that fli4l has to handle daylight saving changes. Because no further information ("+x" or "-x") is given, the clock is adjusted forward one hour when reaching the daylight saving time.
- *M3.5.0,M10.5.0/3:* This means that the change to the daylight saving time occurs on the last Sunday in March at two o'clock and that the change to standard time occurs on the last Sunday in October at three o'clock.

Normally you do not have to touch these settings, unless your fli4l router resides in another time zone. In this case you have to adjust these settings accordingly. In order to do this properly, it is helpful to take a look at the specification of the TZ environment variable which can be found at the following URL:

http://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap08.html

KERNEL_VERSION Chooses the version of the kernel to be used. According to the contents of this variable, the kernel and the kernel modules are selected from *img/kernel-<kernel version>.<compression extension>* and *opt/files/lib/modules/<kernel version>*, respectively. You can also use the suffix "-virt" for newer kernels which activates support for virtual machines. However, these versions require at least a Pentium processor with support for PAE.

KERNEL_BOOT_OPTION Default Setting: `KERNEL_BOOT_OPTION=""`

The contents of this variable is appended to the kernel's command line defined in syslinux.cfg. Some systems require 'reboot=bios' for proper rebooting, i.e. WRAP systems.

COMP_TYPE_ROOTFS Default setting: `COMP_TYPE_ROOTFS='xz'`

This variable selects the compression method to be used for the RootFS archive. Possible values are 'xz', 'lzma', and 'bzip2'.

COMP_TYPE_OPT Default setting: `COMP_TYPE_OPT='xz'`

This variable selects the compression method to be used for the OPT archive. Possible values are 'xz', 'lzma', and 'bzip2'.

POWERMANAGEMENT Default Setting: `POWERMANAGEMENT='acpi'`

The kernel supports different flavours of power management: the somewhat aged APM and the newer ACPI. This variable lets you choose which flavour is to be used. Possible values are 'none' (no power management), 'acpi', and the two APM variants 'apm' and 'apm_rm'. The latter uses a special processor mode before switching the router off.

FLI4L_UUID Default Setting: `FLI4L_UUID="`

This variable contains an universally unique identifier (UUID) which is used to point to a place where persistent data can be stored, e.g. on a USB stick. The UUID can be generated on any Linux system (e.g. on the fli4l router) by executing `'cat␣/proc/sys/kernel/random/uuid'`. Each execution of this command above produces a new UUID which you can use in `FLI4L_UUID` variable. If you create a directory on a persistent medium by the name of this UUID, this directory will be used to store configuration changes as well as persistent run-time data (e.g. DHCP leases). However, the corresponding packages has to support this persistence mechanism (see the documentation to check this). Typically, use 'auto' for the according storage location, instead of a hard-coded path.

If fli4l already stored data using this mechanism before configuring an UUID and creating the directory, this data can be found under /boot/persistent. In this case, you will have to manually move the data to the new location. We advice that you generate and configure the UUID at the very beginning, avoiding the migration later on.

Additionally, please note that `MOUNT_BOOT`='rw' is needed if the storage directory is located on the /boot partition.

We suggest using the /data partition (with the UUID-named directory being a top-level directory there) or an USB stick for the storage location of persistent configuration and run-time data. The file systems allowed are VFAT or, if you use OPT_HD all read-writable filesystems supported there.

IP_CONNTRACK_MAX Default Setting: `IP_CONNTRACK_MAX="`

This variable enables you to change the maximum number of simultaneously existing connections. Normally, a sensible value for this setting is computed automatically, based on the amount of your router's physical RAM. Table 3.2 shows the defaults used.

If you use file sharing programs behind or on the router and your router has only little RAM, you will hit the maximum number of simultaneous connections fastly. This will prevent further connections to be established.
This causes error messages as

```
ip_conntrack: table full, dropping packet
```

Table 3.2.: Automtically generated maximum number of simultaneous connections

RAM in MiB	simultaneous connections
16	1024
24	1280
32	2048
64	4096
128	8192

or

```
ip_conntrack: Maximum limit of XXX entries exceeded
```

The variable IP_CONNTRACK_MAX changes the maximum number of simultaneously existing connections to a fixed value. Each possible connection consumes 350 bytes of RAM, which cannot be used for other things. If you e.g. choose the value '10000', you reserve about 3,34 MB RAM that are lost for any other usage (kernel, RAM disks, programs).

If your router has 32 MiB RAM, it should not be much of a problem to reserve 2 or 3 MiB for the ip_conntrack table. If only 16 MiB RAM or less are available you should be more conservative to prevent your router from running out of RAM.

The setting currently being used can be display on the console by executing

```
cat /proc/sys/net/ipv4/ip_conntrack_max
```

and can be set on-the-fly by executing

```
echo "XXX" > /proc/sys/net/ipv4/ip_conntrack_max
```

where XXX denotes the number of entries. The entries of the IP_CONNTRACK table can be displayed on the console by executing

```
cat /proc/net/ip_conntrack
```

and can be counted by executing

```
cat /proc/net/ip_conntrack | grep -c use
```

LOCALE Default setting: LOCALE='de'

Meanwhile, some fli4l components support multiple languages, for example the console menu and the Web GUI. This variable lets you choose your preferred language. In addition, some components support a private setting to override this global setting if necessary. English is used as a fallback if the language chosen is not supported for some component.

KEYBOARD_LOCALE='auto' tries to find a keyboard layout that is compatible with the LOCALE setting.

By now, the following values are possible: de, en, fr.

3.3. Console settings

CONSOLE_BLANK_TIME Defaut Setting: `CONSOLE_BLANK_TIME="`

Typically, the Linux kernel activates the console's screen saver after some time without console input activity. The variable `CONSOLE_BLANK_TIME` allows you to configure the timeout to be used or to disable the screen saver completely (`CONSOLE_BLANK_TIME=’0’`).

BEEP Defaut Setting: `BEEP=’yes’`

Causes a beep at the end of the boot or shutdown process.

If you enter ‘yes’ here, there will be a beep at the end of the boot or shutdown process. If you suffer from an extreme shortage of space on your boot media or if you don't like your router to beep, use ‘no’ instead.

SER_CONSOLE Default Setting: `SER_CONSOLE=’no’`

The fli4l router can be used completely without any keyboard or graphics adapter. In order to see all kernel boot messages without Telnet or SS access to the router it is possible to redirect all input and output from/to a serial port by setting `SER_CONSOLE`, `SER_CONSOLE_IF`, and `SER_CONSOLE_RATE` to appropriate values.

The serial console can be set to three modes:

`SER_CONSOLE`	console input/output
no	Input and output over tty0
yes	Input and output over serial console
primary	Input and output over serial console, kernel output additionally redirected to tty0
secondary	Input and output over tty0, kernel output additionally redirected to serial console

Changing the value of `SER_CONSOLE` affects the router only if you also update your boot media or if you perform a remote update of the syslinux.cfg file.

If you disable the serial console, please pay attention to enable an alternative access to the router (ssh or physical console).

You will find further information in the appendix under Serial console (Page 338).

SER_CONSOLE_IF Default Setting: `SER_CONSOLE_IF=’0’`

Number of the serial port to use for console output.

This variable controls the number of the serial port to use. Zero corresponds to ttyS0 or COM1 in the DOS world, respectively.

SER_CONSOLE_RATE Default Setting: `SER_CONSOLE_RATE=’9600’`

Transmission rate of the serial port for console output.

This variable contains the Baud rate to use for transmitting data over the serial port. Reasonable values are: 4800, 9600, 19200, 38400, 57600, 115200

3.4. Hints To Identify Problems And Errors

fli4l logs all output produced while booting into the file (*/var/tmp/boot.log*). After the boot process has finished you can review this file at the console or using the web interface.

Sometimes it is useful to generate a more detailed trace of the start sequence, e.g. to analyze the boot process in case of problems. The variable `DEBUG_STARTUP` exists for this very reason. Other settings help developers to find bugs in certain situations; these settings are also documented in this section.

DEBUG_STARTUP Default Setting: `DEBUG_STARTUP='no'`

If set to 'yes', each command to be executed is written to the console while booting. As a change in syslinux.cfg is necessary for enabling this functionality, everything mentioned for `SER_CONSOLE` also applies to this case. If you want to adapt syslinux.cfg by hand, you need to insert `fli4ldebug=yes` to it. Nevertheless, `DEBUG_STARTUP` needs to be set to 'yes'.

DEBUG_MODULES Default Setting: `DEBUG_MODULES='no'`

Some modules are loaded automatically by the kernel without further notification. `DEBUG_MODULES='yes'` activates a mode showing the sequence of all modules being loaded, regardless whether they are loaded explicitly by a script or automatically by the kernel.

DEBUG_ENABLE_CORE Default Setting: `DEBUG_ENABLE_CORE='no'`

If this setting is activated, every program crash causes the creation of a so-called "core dump", a memory image of the process just before the crash. These files are saved in the directory `/var/log/dumps` on the router and can be helpful in finding program errors. More details details can be found in the section "Debugging programs on the fli4l" (Page 243) in the documentation of the SRC package.

DEBUG_MDEV Default Setting: `DEBUG_MDEV='no'`

With `DEBUG_MDEV='yes'` all actions related to the `mdev` daemon will be logged, in detail all additions or removals of device nodes in `/dev` or the loading of firmware. Output is directed to the file `/dev/mdev.log`.

DEBUG_IPTABLES Default Setting: `DEBUG_IPTABLES='no'`

With `DEBUG_IPTABLES='yes'` all `iptables` invocations are logged to `/var/log/iptables.log`, including the return values.

DEBUG_IP Default Setting: `DEBUG_IP='no'`

With `DEBUG_IP='yes'` all invocations of the program `/sbin/ip` are logged to the file `/var/log/wrapper.log`.

3.5. Usage of a customized /etc/inittab

It is possible to let the "init" process start additional programs on additional consoles or to change the default commands. An inittab entry is structured as follows:

```
device:runlevel:action:command
```

The *device* denotes the terminal used for program input/output. Possible devices are terminals tty1-tty4 or serial terminals ttyS0-ttySn with $n <$ the number of available serial ports.

The possible *action*s are typically *askfirst* or *respawn*. Using askfirst lets "init" wait for a keypress before running that command. The respawn action causes the command to be automatically restarted whenever it terminates.

command specifies the program to execute. You have to use a fully qualified path.

The documentation of the Busybox toolkit at http://www.busybox.net contains a detailed description of the inittab format.

The normal inittab file is as follows:

```
::sysinit:/etc/rc
::respawn:cttyhack /usr/local/bin/mini-login
::ctrlaltdel:/sbin/reboot
::shutdown:/etc/rc0
::restart:/sbin/init
```

You could e.g. extend it by the entry

```
tty2::askfirst:cttyhack /usr/local/bin/mini-login
```

in order to get a second login process on the second terminal. To achieve this, simply copy the file opt/etc/inittab to <config directory>/etc/inittab and edit the copy accordingly.

3.6. Localized keyboard layouts

KEYBOARD_LOCALE Default Setting: `KEYBOARD_LOCALE='auto'`

If you sometimes work directly at the router's console you will appreciate a localized keyboard layout. With `KEYBOARD_LOCALE='auto'`, fli4l tries to find a keyboard layout that is compatible with the `LOCALE` setting. With `KEYBOARD_LOCALE=''`, no keyboard layout will be installed on the fli4l router, causing the kernel's default layout to be used. Alternatively, you may set the variable to the name of a local keyboard layout map. If you e.g. use `KEYBOARD_LOCALE='de-latin1'`, the build process checks whether there is a file named de-latin1.map in the directory opt/etc. If this is the case, this file will be used when configuring the keyboard layout.

OPT_MAKEKBL Default Setting: `OPT_MAKEKBL='no'`

If you want to create a map file for your keyboard, you have to proceed as follows:

- Set `OPT_MAKEKBL` to 'yes'.
- Invoke 'makekbl.sh' on the router. Preferably, you use a SSH connection as the keyboard layout changes and this can be quite annoying.
- Follow the instructions.
- You will find your new <locale>.map file in /tmp.

  The tasks to be done directly on the router are now completed.

- Copy the keyboard layout map you have just created to your fli4l directory under opt/etc/<locale>.map. If you now set KEYBOARD_LOCALE='<locale>', your freshly created keyboard layout will be used when building the fli4l images the next time.
- Don't forget to set OPT_MAKEKBL to 'no' again.

3.7. Ethernet network adapter drivers

NET_DRV_N Number of needed network adapter drivers.

If you use the router with ISDN, you typically have one network adapter only, hence the default value is 1. However, if you use DSL, you will have two network adapters in the majority of cases.

You have to separate two cases:

1. Both adapters are of the same type. Then you will have to specify only one driver communicating with both adapters, hence NET_DRV_N='1'.
2. The types of the adapters used differ. Then you have to set the variable to '2' and to configure the drivers separately for both adapters.

NET_DRV_x NET_DRV_x_OPTION This variable contains the name of the driver to be used for the network adapter. The default for NET_DRV_1 is to load the driver for a NE2000 compatible network adapter. More available drivers for a large amount of families of network adapters are included in the tables 3.3 and 3.4.

The 3COM EtherLinkIII network adapter (3c509) has to be configured by the DOS tool 3c509cfg.exe, available under:

ftp://ftp.ihg.uni-duisburg.de/Hardware/3com/3C5x9n/3C5X9CFG.EXE

It should be used for setting the IRQ and I/O port and, if necessary, the type of connection (BNC/TP). The entry NET_DRV_x_OPTION=" can normally be left empty.

Some ISA adapters require the driver to have additional information in order to find the adapter, e.g. the I/O address. This is the case e.g. for NE2000 compatible ISA adapters and the EtherExpress16.

In such a case, you can set

```
NET_DRV_x_OPTION='io=0x340'
```

(or the corresponding numerical value).

If no options are required, you can leave this variable empty.

If you need to specify more than one option, you have to separate them by blanks, e.g.

```
NET_DRV_x_OPTION='irq=9 io=0x340'
```

If you use two identical network adapters, e.g. NE2000 ISA adapters, you have to separate the different I/O ports by commas:

```
NET_DRV_x_OPTION='io=0x240,0x300'
```

No space is allowed before or after the comma!

This does not work with all network adapter drivers. Some of them need to be loaded twice, i.e. you have to use NET_DRV_N='2'. In this case you will have to assign different names to the adapters by using the "-o" option, e.g.

```
NET_DRV_N='2'
NET_DRV_1='3c503'
NET_DRV_1_OPTION='-o 3c503-0 io=0x280'
NET_DRV_2='3c503'
NET_DRV_2_OPTION='-o 3c503-1 io=0x300'
```

We recommend to try the "comma" method first before falling back to loading the driver multiple times.

Some more examples:

- 1 x NE2000 ISA

  ```
  NET_DRV_1='ne'
  NET_DRV_1_OPTION='io=0x340'
  ```

- 1 x 3COM EtherLinkIII (3c509)

  ```
  NET_DRV_1='3c509'
  NET_DRV_1_OPTION=''
  ```

  For this adapter, see also:

  http://extern.fli4l.de/fli4l_faqengine/faq.php?display=faq&faqnr=132&catnr=7&prog=1
  http://extern.fli4l.de/fli4l_faqengine/faq.php?display=faq&faqnr=133&catnr=7&prog=1
  http://extern.fli4l.de/fli4l_faqengine/faq.php?display=faq&faqnr=135&catnr=7&prog=1

- 2 x NE2000 ISA

  ```
  NET_DRV_1='ne'
  NET_DRV_1_OPTION='io=0x320,0x340'
  ```

  Here, you will often need to specify the IRQ values:

  ```
  NET_DRV_1_OPTION='io=0x320,0x340 irq=3,5'
  ```

  You should first try the configuration without specifying any IRQs and enter IRQs only if the network adapters are not found otherwise.

- 2 x NE2000 PCI

  ```
  NET_DRV_1='ne2k-pci'
  NET_DRV_1_OPTION=''
  ```

- 1 x NE2000 ISA, 1 x NE2000 PCI

  ```
  NET_DRV_1='ne'
  NET_DRV_1_OPTION='io=0x340'
  NET_DRV_2='ne2k-pci'
  NET_DRV_2_OPTION=''
  ```

- 1 x SMC WD8013, 1 x NE2000 ISA

```
NET_DRV_1='wd'
NET_DRV_1_OPTION='io=0x270'
NET_DRV_2='ne2k'
NET_DRV_2_OPTION='io=0x240'
```

You can find complete lists of available drivers in the Table of available network adapter drivers and in the Table of available WLAN adapter drivers.

If you need a dummy device, use 'dummy' as your `NET_DRV_x` *and* `IP_NET_x_DEV` *(Page 39)='dummy<number>' as your device.*

Kernel 3.14					Kernel 3.18	
	v	n	vn		v	n
x	x	x	x	isa	3c509	3Com Etherlink III (3c509, 3c509B, 3c5
x	x	x	x	isa	3c515	3Com 3c515 Corkscrev
x	x	x	x	pcmcia	3c574_cs	3Com 3c574 series PCMCIA
x	x	x	x	pcmcia	3c589_cs	3Com 3c589 series PCMCIA
x	x	x	x	pci	3c59x	3Com 3c59x/3c9xx ether
x	x	x	x	pci	8139cp	RealTek RTL-8139C+ series 10/100
x	x	x	x	pci	8139too	RealTek RTL-8139 Fast Et
x	x	x	x	pci	acenic	AceNIC/3C985/GA620 Gigabit
x	x	x	x	pci	alx	Qualcomm Atheros(R) AR816x/AR817x PC
x	x	x	x	pci	amd8111e	AMD8111 based 10/100 Etherne
x	x	x	x	usb	asix	ASIX AX8817X based USB 2.0 Eth
x	x	x	x	pci	atl1	Atheros L1 Gigabit Ether
x	x	x	x	pci	atl1c	Qualcom Atheros 100/1000M Ethe
x	x	x	x	pci	atl1e	Atheros 1000M Ethernet Ne
x	x	x	x	pci	atl2	Atheros Fast Ethernet Net
x	x	x	x	isa	atp	RealTek RTL8002/8012 parallel p
x	x	x	x	usb	ax88179_178a	ASIX AX88179/178A based USB 3.0/2.0 Gi
x	x	x	x	pcmcia	axnet_cs	Asix AX88190 PCMCIA et
x	x	x	x	pci	b44	Broadcom 44xx/47xx 10/100 PC
x	x	x	x	pci	be2net	Emulex OneConnect NIC Driver
x	x	x	x	pci	bna	Brocade 10G PCIe Ether
x	x	x	x	pci	bnx2	Broadcom NetXtreme II BCM5706/5
x	x	x	x	pci	bnx2x	Broadcom NetXtreme II BCM57710/ 57711/ 57711E/ 57712/ 57712_MF/ 578
x	x	x	x	pci	cassini	Sun Cassini(+) etherne
x	x	x	x	usb	catc	CATC EL1210A NetMate USB
x	x	x	x	usb	cdc_eem	USB CDC EEM
x	x	x	x	usb	cdc_ether	USB CDC Ethernet devi
x	x	x	x	usb	cdc_mbim	USB CDC MBIM hos
x	x	x	x	usb	cdc_ncm	USB CDC NCM host
x	x	x	x	usb	cdc_subset	Simple 'CDC Subset' USB netwo
x	x	x	x	usb	cx82310_eth	Conexant CX82310-based ADSL rout
x	x	x	x	pci	cxgb	Chelsio 10Gb Etherne
x	x	x	x	pci	cxgb3	Chelsio T3 Network
x	x	x	x	pci	cxgb4	Chelsio T4/T5 Networ
x	x	x	x	pci	cxgb4vf	Chelsio T4/T5 Virtual Function (
x	x	x	x	pci	de2104x	Intel/Digital 21040/1 series PC
x	x	x	x	isa	de4x5	Digital DE425, DE434, DE435, DI
x	x	x	x	pci	defxx	DEC FDDIcontroller TC/EISA/PCI (DEFTA/DEFEA
x	x	x	x	pci	dl2k	D-Link DL2000-based Gigabit Ethe
x	x	x	x	usb	dm9601	Davicom DM96xx USB 10/100 eth
x	x	x	x	pci	dmfe	Davicom DM910X fast eth
x	x	x	x	virtual	dummy	Dummy Network Interfa

Kernel						
3.14						3.18
	v	n	vn		v	n
x	x	x	x	pci	e100	Intel(R) PRO/100 Netw
x	x	x	x	pci	e1000	Intel(R) PRO/1000 Netw
x	x	x	x	pci	e1000e	Intel(R) PRO/1000 Netw
x	x	x	x	pci	enic	Cisco VIC Ethernet NI
x	x	x	x	pci	epic100	SMC 83c170 EPIC series Et
x	x	x	x	pci	fealnx	Myson MTD-8xx 100/10M Etherne
x	x	x	x	pcmcia	fmvj18x_cs	fmvj18x and compatible PCMCI
x	x	x	x	pci	forcedeth	Reverse Engineered nForce e
x	x	x	x	usb	gl620a	GL620-USB-A Host-to-Host Li
x	x	x	x	pci	hamachi	Packet Engines 'Hamachi' GNIC-II G
x	x	x	x	pci	hp100	HP CASCADE Architecture Driver for 100VG-
x	x	x	x	usb	hso	USB High Speed Optic
x	x	x	x	usb	huawei_cdc_ncm	USB CDC NCM host driver with encapsul
x	x	x	x	pci	i40e	Intel(R) Ethernet Connection XL
x	x	x	x	pci	i40evf	Intel(R) XL710 X710 Virtual Func
x	x	x	x	pci	igb	Intel(R) Gigabit Ethernet N
x	x	x	x	pci	igbvf	Intel(R) Gigabit Virtual Functio
x	x	x	x	usb	int51x1	Intellon usb powerline ada
x	x	x	x	pci	ipg	IC Plus IP1000 Gigabit Ethernet A
x	x	x	x	usb	ipheth	Apple iPhone USB Ether
x	x	x	x	pci	ixgb	Intel(R) PRO/10GbE Net
x	x	x	x	pci	ixgbe	Intel(R) 10 Gigabit PCI Expres
x	x	x	x	pci	ixgbevf	Intel(R) 82599 Virtual Fun
x	x	x	x	pci	jme	JMicron JMC2x0 PCI Express
x	x	x	x	usb	kalmia	Samsung Kalmia USB net
x	x	x	x	usb	kaweth	KL5USB101 USB Ether
x	x	x	x	pci	ksz884x	KSZ8841/2 PCI networ
x	x	x	x	isa	lance	AMD LANCE and PCnet (AT15
x	x	x	x	usb	lg-vl600	LG-VL600 modem's etherne
x	x	x	x	usb	mcs7830	USB to network adapter MC
x	x	x	x	pci	mlx4_core	Mellanox ConnectX HCA lo
x	x	x	x	pci	myri10ge	Myricom 10G driver (10G
x	x	x	x	pci	natsemi	National Semiconductor DP8381x seri
x	x	x	x	isa	ne	NE1000/NE2000 ISA/PnP E
x	x	x	x	pci	ne2k-pci	PCI NE2000 clone
x	x	x	x	usb	net1080	NetChip 1080 based USB Host-to
x	x	x	x	pci	netxen_nic	QLogic/NetXen (1/10) GbE Intelli
x	x	x	x	isa	ni65	AMD Lance Am7990
x	x	x	x	pci	niu	Sun Neptun Ethernet
x	x	x	x	pcmcia	nmclan_cs	New Media PCMCIA ethe
x	x	x	x	pci	ns83820	National Semiconductor DP83820
x	x	x	x	pci	pch_gbe	EG20T PCH Gigabit ethe
x	x	x	x	pci	pcnet32	PCnet32 and PCnetPCI based
x	x	x	x	pcmcia	pcnet_cs	NE2000 compatible PCMCIA
x	x	x	x	usb	pegasus	Pegasus/Pegasus II USB Et
x	x	x	x	usb	plusb	Prolific PL-2301/2302/25A1 USB Ho
x	x	x	x	pci	qla3xxx	QLogic ISP3XXX Network Driver
x	x	x	x	pci	qlcnic	QLogic 1/10 GbE Converged/Intell
x	x	x	x	pci	qlge	QLogic 10 Gigabit PCI-E Et
x	x	x	x	usb	qmi_wwan	Qualcomm MSM Interface (QM
x	x	x	x	pci	r6040	RDC R6040 NAPI PCI FastE
x	x	x	x	usb	r8152	Realtek RTL8152/RTL8153 Based USB
x	x	x	x	pci	r8169	RealTek RTL-8169 Gigabit E
x	x	x	x	usb	rndis_host	USB Host side RNDIS
x	x	x	x	usb	rtl8150	rtl8150 based usb-ether

Kernel 3.14						Kernel 3.18
	v	n	vn		v	n
x	x	x	x	pci	s2io	Neterion 10GbE Server N
x	x	x	x	isa	sb1000	General Instruments SB1
x	x	x	x	pci	sc92031	Silan SC92031 PCI Fast Etherne
x	x	x	x	pci	sfc	Solarflare Communications n
x	x	x	x	pci	sis190	SiS sis190/191 Gigabit Eth
x	x	x	x	pci	sis900	SiS 900 PCI Fast Ethern
x	x	x	x	pci	skfp	SysKonnect FDDI PCI ad
x	x	x	x	pci	skge	SysKonnect Gigabit Ethe
x	x	x	x	pci	sky2	Marvell Yukon 2 Gigabit Et
x	x	x	x	isa	smc-ultra	SMC Ultra/EtherEZ ISA/PnP
x	x	x	x	isa	smc9194	SMC's 9000 series of Etherne
x	x	x	x	pcmcia	smc91c92_cs	SMC 91c92 series PCMCIA e
x	x	x	x	usb	smsc75xx	SMSC75XX USB 2.0 Gigabit Ethe
x	x	x	x	pci	smsc9420	SMSC LAN9420
x	x	x	x	usb	smsc95xx	SMSC95XX USB 2.0 Ethernet
x	x	x	x	usb	sr9700	SR9700 one chip USB 1.1 USB to Ethernet device fro
x	x	x	x	usb	sr9800	SR9800 USB 2.0 USB2NET Dev : http://
x	x	x	x	pci	starfire	Adaptec Starfire Ethern
x	x	x	x	pci	stmmac	STMMAC 10/100/1000 Ethern
x	x	x	x	pci	sundance	Sundance Alta Etherne
x	x	x	x	pci	sungem	Sun GEM Gbit etherne
x	x	x	x	pci	sunhme	Sun HappyMealEthernet(HME) 10/10
x	x	x	x	pci	tehuti	Tehuti Networks(R) Netw
x	x	x	x	pci	tg3	Broadcom Tigon3 ethern
x	x	x	x	pci	tlan	TI ThunderLAN based ethernet P
x	x	x	x	pci	tulip	Digital 21*4* Tulip ether
x	x	x	x	pci	typhoon	3Com Typhoon Family (3C990, 3CR9
x	x	x	x	pci	uli526x	ULi M5261/M5263 fast eth
x	x	x	x	pci	via-rhine	VIA Rhine PCI Fast Ethe
x	x	x	x	pci	via-velocity	VIA Networking Velocity Family Gigabi
		x	x	virtio	virtio_net	Virtio network
		x	x	pci	vmxnet3	VMware vmxnet3 virtual
x	x	x	x	pci	vxge	Neterion's X3100 Series 10GbE PCIe I/OVir
x	x	x	x	isa	wd	Western Digital wd8003/wd8013 ; SMC E
x	x	x	x	pci	winbond-840	Winbond W89c840 Ether
		x	x	xen	xen-netfront	Xen virtual network device f
x	x	x	x	pcmcia	xirc2ps_cs	Xircom PCMCIA ethern
x	x	x	x	pci	xircom_cb	Xircom Cardbus ethern
x	x	x	x	pci	yellowfin	Packet Engines Yellowfin G-NIC Gi
x	x	x	x	usb	zaurus	Sharp Zaurus PDA, and compatil

Table 3.3.: Table of available network adapter driver

Kernel 3.14						Kernel 3.18
	v	n	vn		v	n
x	x	x	x	pci	adm8211	IEEE 802.11b wireless cards based on ADMtek ADM8211
x	x	x	x	isa,pci	airo	Cisco/Aironet 802.11 wireless ethernet cards
x	x	x	x	pcmcia	airo_cs	Cisco/Aironet 802.11 wireless ethernet cards

Kernel						
3.14				3.18		
	v	n	vn		v	n
x	x	x	x	usb	ar5523	Atheros AR5523 based USB dongles
x	x	x	x	usb	at76c50x-usb	Atmel at76x USB Wireless LAN
x	x	x	x	pci	ath10k_pci	Driver support for Atheros QCA988X PCIe devices
x	x	x	x	pci	ath5k	5xxx series of Atheros 802.11 wireless LAN cards
x	x	x	x	usb	ath6kl_usb	Driver support for Atheros AR600x USB devices
x	x	x	x	pci	ath9k	Atheros 802.11n wireless LAN cards
x	x	x	x	usb	ath9k_htc	Atheros driver 802.11n HTC based wireless devices
x	x	x	x	pcmcia	atmel_cs	Atmel at76c50x 802.11 wireless ethernet cards
x	x	x	x	pci	atmel_pci	Atmel at76c50x 802.11 wireless ethernet cards
x	x	x	x	pci,pcmcia	b43	Broadcom B43 wireless
x	x	x	x	pci	b43legacy	Broadcom B43legacy wireless
x	x	x	x	usb	brcmfmac	Broadcom 802.11 wireless LAN fullmac
x	x	x	x	pci	brcmsmac	Broadcom 802.11n wireless LAN
x	x	x	x	usb	carl9170	Atheros AR9170 802.11n USB wireless
x	x	x	x	pcmcia	hostap_cs	Intersil Prism2-based 802.11 wireless LAN cards (PC Card)
x	x	x	x	pci	hostap_pci	Intersil Prism2.5-based 802.11 wireless LAN PCI cards
x	x	x	x	pci	hostap_plx	Intersil Prism2-based 802.11 wireless LAN cards (PLX)
x	x	x	x	pci	ipw2100	Intel(R) PRO/Wireless 2100 Network
x	x	x	x	pci	ipw2200	Intel(R) PRO/Wireless 2200/2915 Network
x	x	x	x	pci	iwl3945	Intel(R) PRO/Wireless 3945ABG/BG Network Connection driver for Linu
x	x	x	x	pci	iwl4965	Intel(R) Wireless WiFi 4965 driver for Linux
x	x	x	x	pci	iwlwifi	Intel(R) Wireless WiFi driver for Linux
x	x	x	x	pcmcia	libertas_cs	Marvell 83xx compact flash WLAN cards
x	x	x	x	usb	libertas_tf_usb	8388 USB WLAN Thinfirm
x	x	x	x	virtual	mac80211_hwsim	Software simulator of 802.11 radio(s) for mac80211
x	x	x	x	pci	mwifiex_pcie	Marvell WiFi-Ex PCI-Express Driver version 1.0
x	x	x	x	usb	mwifiex_usb	Marvell WiFi-Ex USB Driver version1.0
x	x	x	x	pci	mwl8k	Marvell TOPDOG(R) 802.11 Wireless Network
x	x	x	x	pcmcia	orinoco_cs	PCMCIA Lucent Orinoco, Prism II based and similar wireless cards
x	x	x	x	pci	orinoco_nortel	wireless LAN cards using the Nortel PCI bridge
x	x	x	x	pci	orinoco_plx	wireless LAN cards using the PLX9052 PCI bridge
x	x	x	x	pci	orinoco_tmd	wireless LAN cards using the TMD7160 PCI bridge
x	x	x	x	usb	orinoco_usb	Orinoco wireless LAN cards using EZUSB bridge
x	x	x	x	pci	p54pci	Prism54 PCI wireless
x	x	x	x	usb	p54usb	Prism54 USB wireless
x	x	x	x	pcmcia	ray_cs	Raylink/WebGear wireless LAN
x	x	x	x	usb	rndis_wlan	RNDIS based USB Wireless adapters
x	x	x	x	pci	rt2400pci	Ralink RT2400 PCI & PCMCIA Wireless LAN
x	x	x	x	pci	rt2500pci	Ralink RT2500 PCI & PCMCIA Wireless LAN
x	x	x	x	usb	rt2500usb	Ralink RT2500 USB Wireless LAN
x	x	x	x	pci	rt2800pci	Ralink RT2800 PCI & PCMCIA Wireless LAN
x	x	x	x	usb	rt2800usb	Ralink RT2800 USB Wireless LAN
x	x	x	x	pci	rt61pci	Ralink RT61 PCI & PCMCIA Wireless LAN
x	x	x	x	usb	rt73usb	Ralink RT73 USB Wireless LAN
x	x	x	x	pci	rtl8180	RTL8180 / RTL8185 PCI wireless
x	x	x	x	usb	rtl8187	RTL8187/RTL8187B USB wireless
x	x	x	x	pci	rtl8188ee	Realtek 8188E 802.11n PCI wireless
x	x	x	x	pci	rtl8192ce	Realtek 8192C/8188C 802.11n PCI wireless
x	x	x	x	usb	rtl8192cu	Realtek 8192C/8188C 802.11n USB wireless
x	x	x	x	pci	rtl8192de	Realtek 8192DE 802.11n Dual Mac PCI wireless
x	x	x	x	pci	rtl8192se	Realtek 8192S/8191S 802.11n PCI wireless
x	x	x	x	pci	rtl8723ae	Realtek 8723E 802.11n PCI wireless
x	x	x	x	usb	sierra_net	USB-to-WWAN Driver for Sierra Wireless modems
x	x	x	x	pcmcia	spectrum_cs	Symbol Spectrum24 Trilogy cards with firmware downloader
x	x	x	x	usb	usb8xxx	8388 USB WLAN

Kernel						
3.14				3.18		
	v	n	vn		v	n
x	x	x	x	pci	wil6210	60g WiFi WIL6210 card
x	x	x	x	pcmcia	wl3501_cs	Planet wl3501 wireless
x	x	x	x	usb	zd1201	ZyDAS ZD1201 based USB Wireless adapters
x	x	x	x	usb	zd1211rw	USB driver for devices with the ZD1211 chip

Table 3.4.: Table of available WLAN adapter drivers; legend: v=virt, n=nonfree, v

3.8. Networks

IP_NET_N Default Setting: `IP_NET_N='1'`

Number of networks to bound to the IP procotol, normally one ('1'). If you set `IP_NET_N` to zero because you don't have any IP networks or because you configure them in a different way, mkfli4l will emit a warning when building the archives. You can disable this warning by using `IGNOREIPNETWARNING`='yes'.

IP_NET_x Default Setting: `IP_NET_1='192.168.6.1/24'`

The IP address and the net mask of the router's n-th device using the CIDR[1] notation. If you want the router to receive its IP address dynamically via a DHCP-client it is possible to set this variable to 'dhcp'.

The following table shows how the CIDR notation and the dot notation for net masks are connected.

CIDR	Net mask	Number of IP addresses
/8	255.0.0.0	16777216
/16	255.255.0.0	65536
/23	255.255.254.0	512
/24	255.255.255.0	256
/25	255.255.255.128	128
/26	255.255.255.192	64
/27	255.255.255.224	32
/28	255.255.255.240	16
/29	255.255.255.248	8
/30	255.255.255.252	4
/31	255.255.255.254	2
/32	255.255.255.255	1

Note: As one address is reserved for the network and one for broadcasting, the maximum number of hosts in the network is computed by: `Number_Hosts = Number_IPs - 2`. Consequently, the smallest possible net mask is `/30` which corresponds to four IP addresses and hence to two possible hosts.

[1] Classless Inter-Domain Routing

IP_NET_x_DEV Default Setting: IP_NET_1_DEV='eth0'

Required: device name of the network adapter.

Starting with version 2.1.8, the name of the device used has to be supplied! Names of network devices typically start with 'eth' followed by a number. The first network adapter recognized by the system receives the name 'eth0', the second one 'eth1' and so on.

Example:

```
IP_NET_1_DEV='eth0'
```

The fli4l router is also able to do IP aliasing, i.e. to assign multiple IP addresses to a single network adapter. Additional IP addresses are simply specified by linking another network to the same device. When mkfli4l checks the configuration you are informed that you define such an alias—you can ignore the warning in this case.

Example:

```
IP_NET_1='192.168.6.1/24'
IP_NET_1_DEV='eth0'
IP_NET_2='192.168.7.1/24'
IP_NET_2_DEV='eth0'
```

IP_NET_x_MAC Default Setting: IP_NET_1_MAC="

Optional: MAC address of the network adapter.

With this variable you are able to change the hardware address (MAC) of your network adapter. This is useful if you want to use a DHCP provider expecting a certain MAC address. If you leave IP_NET_x_MAC empty or remove the variable definition completely, the original MAC address of your network adapter will be used. Most users will never need to use this variable.

Example:

```
IP_NET_1_MAC='01:81:42:C2:C3:10'
```

IP_NET_x_NAME Default Setting: IP_NET_x_NAME="

Optional: Assigning a name to the IP address of a network adapter.

If you perform reverse DNS lookup of the network adapter's IP address, the result is typically a name like 'fli4l-ethx.<domain>'. You can use the variable IP_NET_x_NAME in order to change that name which will be returned when performing reverse DNS lookup. If the IP address is globally accessible, you can use this setting to enforce that reverse DNS lookups always return a globally accessible name.

Example:

```
IP_NET_2='80.126.238.229/32'
IP_NET_2_NAME='ajv.xs4all.nl'
```

IP_NET_x_TYPE

IP_NET_x_COMMENT Default Setting: `IP_NET_x_COMMENT=”`

Optional: You can use this setting to assign a ‘meaningful’ name to a network device. This name can then be used in packages like rrdtool for identifying the network.

3.9. Additional routes (optional)

IP_ROUTE_N Number of additional network routes. Additional network routes are mandatory if e.g. other routers in the LAN exist which have to be used to access other networks.

Normally, you do not need to specify any other network routes.

Default setting: `IP_ROUTE_N=’0’`

IP_ROUTE_x The additional routes `IP_ROUTE_1`, `IP_ROUTE_2`, ... are structured as follows:

```
network/netmaskbits gateway
```

In this case, `network` is the network address, `/netmaskbits` the net mask using the CIDR (Page 38) notation and `gateway` the address of the router needed for accessing the other network. Obviously, the gateway and the fli4l router are required to be in the same network! For example, if the network 192.168.7.0 with net mask 55.255.255.0 can be accessed through the gateway 192.168.6.99 you have to add the following entry:

```
IP_ROUTE_N='1'
IP_ROUTE_1='192.168.7.0/24 192.168.6.99'
```

If you use the fli4l router as a pure Ethernet router and not for routing Internet traffic, you can use some IP_ROUTE_x variable for specifying a default route. In order to achieve this, you have to specify ‘0.0.0.0/0’ for ‘network/netmaskbits’, as can be seen in the following example.

```
IP_ROUTE_N='3'
IP_ROUTE_1='192.168.1.0/24 192.168.6.1'
IP_ROUTE_2='10.73.0.0/16 192.168.6.1'
IP_ROUTE_3='0.0.0.0/0 192.168.6.99'
```

3.10. The Packet Filter

The Linux kernel used by fli4l provides a packet filter which controls who is allowed to communicate with or through the Router. Furthermore, things like port forwarding (a packet addressed to the router is forwarded to another internal computer) and masquerading (packets

sent from a computer behind the router are changed to look as if they came from the router itself) can be realized.

The structure of the packet filter is shown in Figure 3.1.

Packets arrive over a network interface and pass through the PREROUTING-chain. Here the packets addressed to the router are passed to another computer by changing destination address and destination port. If the packet is addressed to the router it is sent to the INPUT-chain, if not, to the FORWARD-chain. Both chains will check if the packet is permitted. If the packet is accepted, it is delivered to the local destination process or passed via the POSTROUTING-chain (in which packet masquerading is done) to the network interface by which it can reach its target. Locally generated packets are filtered in the OUTPUT-chain and finally (if successfully) also pass through the POSTROUTING-chain to the correct network interface.

With the packet filter configuration, the individual chains of the packet filter can be modified directly. An individual array exists for each chain, one for the INPUT-chain (PF_INPUT_%), one for the FORWARD-chain (PF_FORWARD_%), one for the OUTPUT-chain (PF_OUTPUT_%), one for the PREROUTING-chain (managing port forwarding) (PF_PREROUTING_%), and one for the POSTROUTING-chain, managing packet masquerading (PF_POSTROUTING_%).

An entry in one of these arrays consists mainly of an action (see below) which can be restricted by additional conditions. These conditions relate to properties of the considered packet. A packet contains information about its origin (source PC that has sent the packet), its target (to which PC and which application should the packet be delivered) and much more. Conditions can refer to the following properties of a packet:

- source (source address, source port or both)
- destination (destination address, destination port or both)
- protocol
- interface on which the packet comes in or goes out
- MAC-address of the originating PC
- state of the packet or the connection the packet comes from

If a packet comes in, the entries resp. the resulting rules generated are processed from top to bottom and the first action to which all conditions apply is performed. If none of the rules matches, the default action is executed, which may be specified for (almost) any table.

An entry has the following format, bearing in mind that all restrictions are optional:

```
restriction{0,} [[source] [destination]] action [BIDIRECTIONAL|LOG|NOLOG]
```

At all points where networks, IP addresses or hosts need to be specified, you can also refer to IP_NET_%, IP_NET_%_IPADDR or via @hostname to a host from HOST_%. If OPT_DNS is enabled, then outside of actions via @fqdn also hosts which are *nicht* mentioned in HOST_% can be referenced by their names. This is particularly useful if dealing with external hosts which also possess many (and changing) IP addresses.

3.10.1. Packet Filter Actions

The following actions appy:

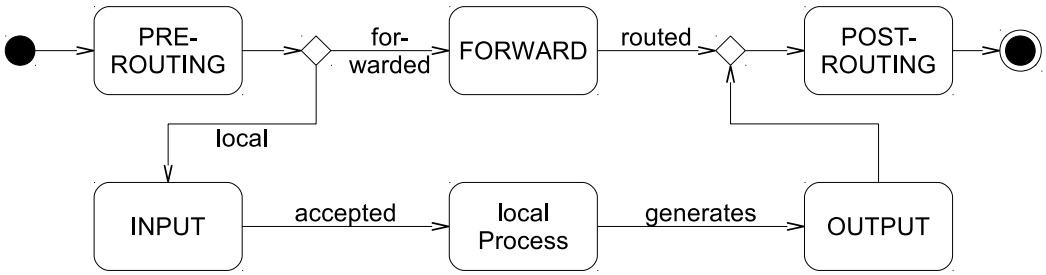


Figure 3.1.: Packet Filter Structure

Action	chain(s)	Meaning
ACCEPT	all	Accept the packet.
DROP	INPUT FORWARD OUTPUT	Drop the packet (the sender recognizes that just because no answer and no error message comes back).
REJECT	INPUT FORWARD OUTPUT	Reject the packet (the sender gets a corresponding error message).
LOG	all	Log the packet and proceed to the next rule. To distinguish log entries a prefix may be used, specified by LOG:log-prefix. The maximum length of this prefix is 28 characters and it may contain letters, numbers, hyphens (-), and underscores (_).
MASQUERADE	POSTROUTING	Mask the packet: Replace the source address of the packet by the own one and make sure that replies for this connection are redirected to the correct computer.
SNAT	POSTROUTING	Replace source address and source port of the packet by the address specified as a parameter for SNAT (for all packets belonging to the connection in consideration).
DNAT	PREROUTING	Replace destination address and destination port of the packet by the address specified as a parameter for SNAT (for all packets belonging to the connection in consideration).
REDIRECT	PREROUTING OUTPUT	Replace destination port of the packet by the address specified as a parameter for SNAT (for all packets belonging to the connection in consideration).
NETMAP	PREROUTING POSTROUTING	Copy destination resp. source address of the packet to the range specified as a parameter for NETMAP; the ports stay unchanged (for all packets belonging to the connection in consideration; while changing the destination address in the PREROUTING-chain and the source address of the POSTROUTING-chain).

Table 3.5.: Packet Filter Actions

Some of these actions may be modified in behaviour by using the options BIDIRECTIONAL, LOG or NOLOG. BIDIRECTIONAL generates the same rule a second time with source and destination adresse exchanged (and source and destination port exchanged and/or in- and outbound network interface exchanged if specified). LOG/NOLOG activates resp. deactivates logging for this rule.

3.10.2. Restrictions For Rules

Restrictions may be defined by constraints explained in the following sections. You may use `any` at any place where you don't want restrictions but want/have to specify something. Constraints can be specified in any order if they have a preceding prefix. This applies to all restrictions, except for specifying a source or destination address which must always be placed directly in front of the action, other constraints must be specified before. Restrictions can also be negated, simply prefix them by a `!`.

Constraints For Source And Target

Each packet contains source and target informations in a tuple of an IP address and ports.[2] This source resp. target can serve as a constraint and may be addressed like this:

Expression	Meaning
`ip`	a simple IP address
`network`	a network declaration in the form of `<ip>/<netmask>`
`port[-port]`	a port resp. a port range
`IP_NET_x_IPADDR`	the IP address of the `x` router's interface
`IP_NET_x`	the `x` router's subnet
`IP_ROUTE_x`	the subnet `x` specified in the route (default routes can't be used, they would match `any` and are excluded precautiously)
`@name`	one of the names or aliases set via HOST_%_*; the associated IP address will be filled in here
`<ip oder netzwerk>:port[-port]`	Host- resp. network address in one of the variants above, combined with a port resp. port range

Table 3.6.: Constraints For Source And Target In Paket Filter Rules

Example: `'192.168.6.2 any DROP'`

If two of these lines shine up the first will be considered as source and the second as target. Hence, in this example we drop the packets originating from the computer with the IP address 192.168.6.2, regardless of where they are targeted.

If only one line exists the decision if target or source is meant will be made depending on the value, which is quite easy:

- If it contains a port value, target is meant,
- in all other cases the source is.

[2] A port only exists for TCP- and UDP-packets.

If you would like to shorten the example above you could write `'192.168.6.2 DROP'`. No port is mentioned, hence the constraint is valid for the source (the machine the packet originated from).

If we were to allow communication with the `ssh`-deamon, we could write `'any any:22 ACCEPT'` (packets from any machine to `ssh`-port 22 of any machine will be accepted) or even shorter `'22 ACCEPT'`. Only a port is mentioned, hence we address the target and thus all packets targeted to port 22.

For simplification you may append BIDIRECTIONAL to the action to express that the rule is valid for both communication directions. Then rules will be generated with source and target addresses and if applicable ports and network interfaces exchanged while leaving the rest untouched.

Examples:

`127.0.0.1 ACCEPT`	local communication (source 127.0.0.1) is allowed
`any 192.168.12.1 DROP`	packets to address 192.168.12.1 will be dropped
`any 192.168.12.1 DROP LOG`	packets to address 192.168.12.1 will be dropped and logged additionally
`any 192.168.12.1 DROP NOLOG`	packets to address 192.168.12.1 will be dropped but not logged
`22 ACCEPT`	packets to port 22 (`ssh`) will be accepted
`IP_NET_1_NET ACCEPT`	packets from the subnet connected to the first interface will be accepted
`IP_NET_1_NET IP_NET_2_NET ACCEPT BIDIRECTIONAL`	communication between the subnets connected to the first and second interface are allowed

Interface Constraints

A rule can be restricted concerning the Interface on which a packet was received resp. will be transmitted. The format is as follows: `if:`*in*`:`*out*

In the `INPUT`-chain the interface for outbound packets is not restrictable (the packet does not leave anyway), in the `POSTROUTING`-chain the interface for received packets is not restrictable, because the informations about it do not exist anymore. Only in the `FORWARD`-chain constraints for both can be defined.

Possible values for *in* resp. *out*:

- `lo` (Loopback-interface, local communication on the router)
- `IP_NET_x_DEV`
- `pppoe` (the PPPoE-interface; only with package `dsl` or `pppoe_server` activated).
- `any`

Protocol Constraints

A rule can be restricted concerning the protocol a packet belongs to. The format is as follows: `prot:`*protocol* resp. `prot:`*icmp*`:`*icmp-type*. *protocol* can be set to one of the following values:

- `tcp`
- `udp`
- `gre` (Generic Routing Encapsulation)
- `icmp` (additionally you can specify a name for the ICMP-type to be filterd (`echo-reply` or `echo-request`), i.e. `prot:icmp:echo-request`)

- numeric value of the protocol-ID (i.e. 41 for IPv6)
- `any`

If such a constraint does not exists, but port numbers should be used in a rule, then the rule is generated *twice*, once for the `tcp` and once for the `udp` protocol.

MAC-Address Constraints

Via `mac:`*mac-address* constraints based on the MAC address may be specified.

Packet State Constraints

fli4l's packet filter gathers informations on the state of connections. This informations can be used to filter packets, i.e let only packets pass that belong to connections already existing. The state of a connection can take this values:[3]

State	Meaning
`INVALID`	The packet does not belong to a know connection.
`ESTABLISHED`	The packet belongs to a connection, where packets have already been transmitted in both directions.
`NEW`	The packet has established a new connection or belongs to a connection that did not have packets transmitted in both directions.
`RELATED`	The packet establishes a new connection, but has a relation to an already existing connection (i.e. `ftp` establishes a separate connection for data transfer).

Table 3.7.: Packet State Constraints in Packet Filter Rules

States are defined as follows: `state:`*state(s)*. If you want to specify more than one state they have to be separated by commas. I.e. to let packets pass that belong directly or indirectly to established connections write `state:ESTABLISHED,RELATED` (this makes sense in `INPUT`- or `FORWARD`-chain).

Constraints Based On The Frequency Of Actions

Under certain circumstances you may wish to restrict the frequency of actions, i.e. allow only one ICMP-Echo request per second. This may be reached with `limit`-constraints, which look like this: `limit:`*Frequency:Burst*. The frequency is specified as *n/time units* (second, minute, hour, day), however, events may also occur in rapid succession (Burst). `limit:3/minute:5` for example means that a maximum of three events per minute is allowed, but also five events in rapid succession will be accepted.

[3] see http://www.sns.ias.edu/~jns/files/iptables_talk/x38.htm for a detailed description

3.10.3. Using Templates With The Packet Filter

To simplify dealing with the packet filter you may summarize rules frequently occuring in templates. Thus, it is possible to provide a wide range of packet filtering rules and combine them in a collection with a symbolic name. Instead of directly using protocols and port numbers, you may then use entries such as `tmpl:ssh` if you want to use the `ssh` protocol in a rule. How to deal with templates is shown here using the example of `ssh`.

If you want to reach your fli4l from the Internet via `ssh`, write into an entry in the array variable `PF_input_%` the corresponding service name (here `ssh`) preceded by `tmpl` and the action to apply for this service. Example:

```
PF_INPUT_2='tmpl:ssh ACCEPT'
```

`tmpl:` means that the rule should be based on a template. Specify the name of the service after the ':', adapted to our example hence `ssh`. At last you have to set an action to be bound to the service. Since we want to acces the fli4l over the internet, we allow the connection with ACCEPT. Restrictions for IP-addresses or nets are not provided so the `ssh`-service will be accessible on all interfaces from all networks. If you want to invoke further restrictions for accessing the `ssh`-service you may use the packet filter notation already explained above.

For which services rules are predefined (e.g. templates exist) can be seen in the template file at `opt/etc/fwrules.tmpl/templates`. A list in a table follows (see table 3.8).

Template	Protocol	Port(s)
dhcp	udp	67-68
dns	tcp/udp	53
elster	tcp	159.154.8.2:21
elster	tcp	159.154.8.35:21
elster	tcp	193.109.238.26:8000
elster	tcp	193.109.238.27:8000
elster	tcp	193.109.238.58:80
elster	tcp	193.109.238.59:80
elster	tcp	62.157.211.58:8000
elster	tcp	62.157.211.59:8000
elster	tcp	62.157.211.60:8000
elster	tcp	80.146.179.2:80
elster	tcp	80.146.179.3:80
ftp	tcp	21
http	tcp	80
https	tcp	443
hylafax	tcp	4559
imap	tcp	143
imaps	tcp	993
imond	tcp	5000
ipmi	tcp	22
ipmi	tcp	2937
ipmi	tcp	443
ipmi	tcp	5120
ipmi	tcp	5123
ipmi	tcp	5900
ipmi	tcp	5901
ipmi	tcp	80
ipmi	tcp	8889
ipmi	udp	623

Template	Protocol	Port(s)
irc	tcp	6667
ldap	tcp/udp	389
mail	tcp	110
mail	tcp	143
mail	tcp	25
mail	tcp	465
mail	tcp	993
mail	tcp	995
mysql	tcp	3306
nfs	tcp/udp	111
nfs	tcp/udp	2049
nntp	tcp	119
ntp	udp	123
oracle	tcp	1521
pcanywhere	tcp	5631-5632
ping	icmp:0	
ping	icmp:8	
pop3	tcp	110
pop3s	tcp	995
privoxy	tcp	8118
proxmox	tcp	8006
proxmox	tcp	5900
proxmox	tcp	3128
rdp	tcp	3389
rsync	tcp	873
samba	tcp	139
samba	tcp	445
samba	udp	137-138
sip	tcp/udp	5060-5061
smtp	tcp	25
snmp	tcp/udp	161
socks	tcp	1080
squid	tcp	3128
ssh	tcp	22
ssmtp	tcp	465
submission	tcp	587
svn	tcp	3690
syslog	udp	514
teamspeak	tcp	14534
teamspeak	tcp	51234
teamspeak	udp	8767
telmond	tcp	5001
telnet	tcp	23
teredo	udp	3544
tftp	udp	69
time	tcp/udp	37
traceroute	udp	33404-33464
vdr	tcp	6419
vnc	tcp	5900
whois	tcp	43
xbl	tcp/udp	3074
xbl	udp	88
xmppclient	tcp	5222
xmppserver	tcp	5269

Template	Protocol	Port(s)

Table 3.8.: Templates Included With fli4l

The Syntax for this kind of packet filter rules is

```
tmpl:<Name of the service> <Constraint> <Action>
```

`<Constraint>` allows everything mentioned at 3.10.2. Possible values for `<Action>` are listed and described in 3.10.1.

Some more examples should clarify the process. At first let's have a look at `PF_PREROUTING`:

```
PF_PREROUTING_N='2'
PF_PREROUTING_1='tmpl:xbl dynamic DNAT:@xbox'
PF_PREROUTING_2='tmpl:https dynamic DNAT:192.168.193.250'
```

The rule `PF_PREROUTING_1` supplies the Xbox with everything necessary for Xbox Live. By the use of `tmpl:xbl` all ports and protocols used for Xbox Live will be forwarded to the `xbox`. Instead of using an IP address we use an entry from the `HOST_%_NAME`-array. `dynamic` tells the fli4l to forward all ports from the internet interface.

The second rule forwards the `https`-protocol to a webserver in a DMZ (Demilitarized Zone).

No let's have a look at `PF_INPUT`:

```
PF_INPUT_N='3'
PF_INPUT_1='if:IP_NET_1_DEV:any ACCEPT'
PF_INPUT_2='if:pppoe:any prot:tcp 113 ACCEPT'
PF_INPUT_3='if:br0:any tmpl:dns @xbox IP_NET_1_IPADDR ACCEPT'
```

The first rule allows access to the router for everyone from the net defined in `IP_NET_1`. The second rule opens the `ident`-port needed for package `oident`. The third rule allows the xbox to access fli4l's DNS server. Notice the use of a host alias here.

`PF_FORWARD` and `PF_POSTROUTING` do not provide `tmpl`-specific content.

It is also possible to create templates yourself or for other packages to provide their own ones. To create a template you only need to create a text file with the rules in it and name it like the template. For a private template file use the directory `etc/fwrules.tmpl` (create it if necessary) under your `config` directory as shown in picture 3.2. Package developers or users needing templates for more than one configuration may place their template files directly in `opt/etc/fwrules.tmpl`. The templates in the user's `config` directory override other settings, though. The templates included in fli4l will be interpreted as the last ones. This enables you to „override“ fli4l's templates when providing templates by the same name in your `config`-directory.

If, for example you like to create the template `vpn_friends`, create a file by the name `vpn_friends`. The template should contain the services `ssh`, `smtp`, `dns` and `samba`. Hence you write the following to `vpn_friends`:

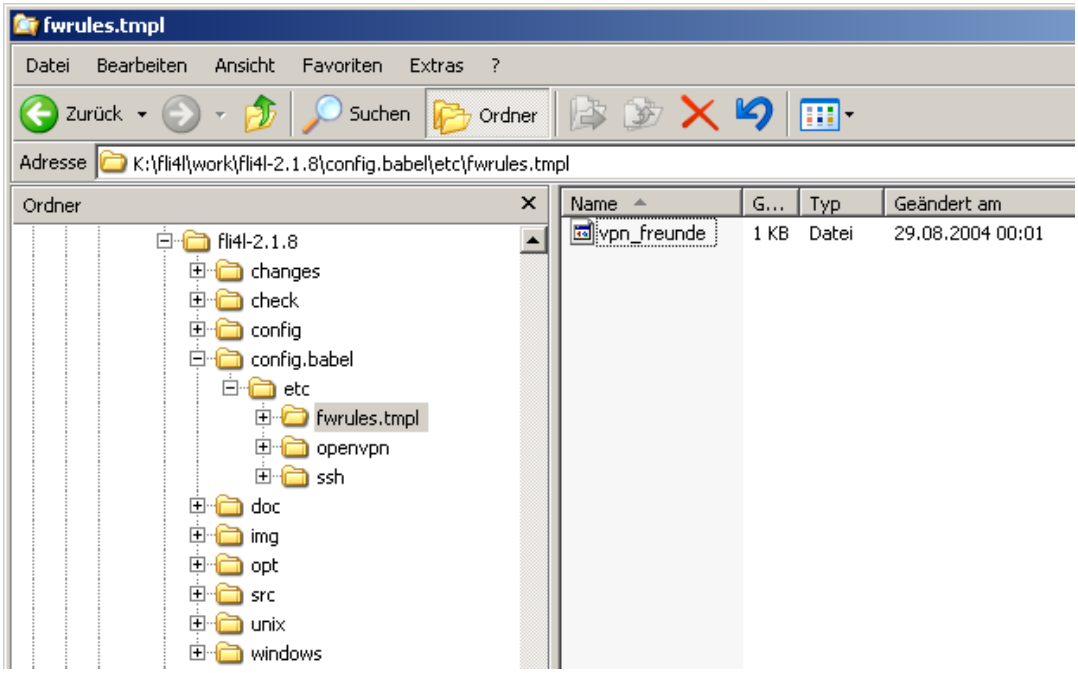


Figure 3.2.: Directory Structure fli4l

```
prot:tcp 22
prot:tcp 25
53
prot:udp 137-138
prot:tcp 139
prot:tcp 445
```

Every time you use the template `vpn_friends` rules will be created for all contained protocols and ports. `PF_FORWARD_x='tmpl:vpn_friends ACCEPT'` will create theses `FORWARD`-rules:

```
prot:tcp 22 ACCEPT
prot:tcp 25 ACCEPT
53 ACCEPT
prot:udp 137-138 ACCEPT
prot:tcp 139 ACCEPT
prot:tcp 445 ACCEPT
```

3.10.4. Configuration Of The Packet Filter

The packet filter is mainly configured by four array-variables:

- `PF_INPUT_%` configures the `INPUT`-chain,
- `PF_FORWARD_%` configures the `FORWARD`-chain,
- `PF_OUTPUT_%` configures the `OUTPUT`-chain,
- `PF_PREROUTING_%` configures the `PREROUTING`-chain and

- `PF_POSTROUTING_%` configures the `POSTROUTING`-chain.

For all chains following applies the setting of the protocol level in `PF_LOG_LEVEL`, which may be set to one of these values: `debug`, `info`, `notice`, `warning`, `err`, `crit`, `alert`, `emerg`.

Then `INPUT`-Chain

The `INPUT`-chain defines who is allowed to access the router. If no rule of the `INPUT`-chain matches, the default action handles the packet and the protocol variable decides wheter a rejection will be written to the system-protocol or not.

The following restrictions apply to the parameters:

- Only ACCEPT, DROP and REJECT can be specified as actions.
- If using interface constraints only the receiving interface can be restricted.

PF_INPUT_POLICY This variable describes the default action to be taken if no other rule applies. Possible values:

- ACCEPT (not recommended)
- REJECT
- DROP (not recommended)

PF_INPUT_ACCEPT_DEF If this variable is set to 'yes' default rules will be generated needed for the correct function of the router. Use 'yes' as a default here.

If you want to configure the router's behaviour completely yourself you may enter 'no' here but you will have to define all rules on your own then. An equivalent to the default behaviour would look like this (the explanation of user defined chains can be found here (Page 54)):

```
PF_INPUT_ACCEPT_DEF='no'
#
# limit ICMP echo requests - use a separate chain
#
PF_USR_CHAIN_N='1'
PF_USR_CHAIN_1_NAME='usr-in-icmp'
PF_USR_CHAIN_1_RULE_N='2'
PF_USR_CHAIN_1_RULE_1='prot:icmp:echo-request length:0-150 limit:1/second:5 ACCEPT'
PF_USR_CHAIN_1_RULE_2='state:RELATED ACCEPT'

PF_INPUT_N='4'
PF_INPUT_1='prot:icmp usr-in-icmp'
PF_INPUT_2='state:ESTABLISHED,RELATED ACCEPT'
PF_INPUT_3='if:lo:any ACCEPT'
PF_INPUT_4='state:NEW 127.0.0.1 DROP BIDIRECTIONAL'
```

The first rule branches to the rate limited "usr-in-icmp"-chain. The second only accepts packets belonging to established connections (packets that have either the state `ESTABLISHED` or `RELATED`), and the third one allows local communication (`if:lo:any ACCEPT`).

The fourth filters packets that pretend to be local communication but are not accepted by the rules defined before.

If you work with OpenVPN, the rules have to be enhanced to enable packets used by the chains there.

```
PF_INPUT_N='5'
...
PF_INPUT_5='ovpn-chain'
```

PF_INPUT_LOG Defines if rejected packets should be logged by the kernel. Log output can be directed to the syslog deamon by activating `OPT_KLOGD`.

PF_INPUT_LOG_LIMIT Defines how often log entries will be generated. The frequency is described as *n/time units* with bursts in analog to the limit constraints, e.g. `3/minute:5`. If this entry is empty a default of `1/second:5` is used, if set to `none`, the limit constraints are disabled.

PF_INPUT_REJ_LIMIT PF_INPUT_UDP_REJ_LIMIT Specifies how often a REJECT-packet is generated when rejecting incoming packets. The frequency is described as *n/time units* with bursts in analog to the limit constraints, e.g. `3/minute:5`. If this entry is empty a default of `1/second:5` is used, if set to `none`, the limit constraints are disabled.

PF_INPUT_ICMP_ECHO_REQ_LIMIT Defines how often fli4l should react to a ICMP-Echo-request.The frequency is described as *n/time units* with bursts in analog to the limit constraints, e.g. `3/minute:5`. If the limit is reached packets will be ignored (DROP). If this entry is empty a default of `1/second:5` is used, if set to `none`, the limit constraints are disabled.

PF_INPUT_ICMP_ECHO_REQ_SIZE Defines the allowed size of an ICMP-Echo-request (in bytes). The packet header has to be included in this setting besides the pure data. The default is 150 bytes.

PF_INPUT_N PF_INPUT_x PF_INPUT_x_COMMENT A list of rules that describe which packets the router should accept resp. reject.

The FORWARD-Chain

By using the FORWARD-chain will be configured which packets are forwarded by the router. If no rule of the FORWARD-chain matches, the default action handles the packet and the protocol variable decides wheter a rejection will be written to the system-protocol or not.

With the used parameters the restriction applies that only the actions ACCEPT, DROP and REJECT are allowed.

PF_FORWARD_POLICY This variable describes the default action to be taken if no other rule applies. Possible values:

- ACCEPT
- REJECT

- DROP

PF_FORWARD_ACCEPT_DEF Determines if the router accepts packets belonging to established connections. If this variable is set to 'yes', fli4l generates a rule for accepting packets of the according state automatically:

`'state:ESTABLISHED,RELATED ACCEPT'`,

aswell as a rule to drop packets of unknown state:

`'state:INVALID DROP'`.

and at last a rule to drop packets with faked IP addresses:

`'state:NEW 127.0.0.1 DROP BIDIRECTIONAL'`.

In addition the other subsystems will generate some default rules – a configuration without default rules with port forwarding and OpenVPN would contain at least the following rules:

```
PF_FORWARD_ACCEPT_DEF='no'
PF_FORWARD_N='5'
PF_FORWARD_1='state:ESTABLISHED,RELATED ACCEPT'
PF_FORWARD_2='state:INVALID DROP'
PF_FORWARD_3='state:NEW 127.0.0.1 DROP BIDIRECTIONAL'
PF_FORWARD_4='pfwaccess-chain'
PF_FORWARD_5='ovpn-chain'
```

PF_FORWARD_LOG Defines if rejected packets should be logged by the kernel. Log output can be directed to the syslog deamon by activating `OPT_KLOGD`.

PF_FORWARD_LOG_LIMIT Defines how often log entries will be generated. The frequency is described as *n/time units* with bursts in analog to the limit constraints, e.g. `3/minute:5`. If this entry is empty a default of `1/second:5` is used, if set to `none`, the limit constraints are disabled.

PF_FORWARD_REJ_LIMIT PF_FORWARD_UDP_REJ_LIMIT Specifies how often a REJECT-packet is generated when rejecting incoming packets. The frequency is described as *n/time units* with bursts in analog to the limit constraints, e.g. `3/minute:5`. If this entry is empty a default of `1/second:5` is used, if set to `none`, the limit constraints are disabled.

PF_FORWARD_N PF_FORWARD_x PF_FORWARD_x_COMMENT A list of rules that describe which packets the router should forward resp. reject.

The `OUTPUT`-Chain

The `OUTPUT`-chain configures what the router is allowed to access. If no rule of the `OUTPUT`-chain matches, the default action handles the packet and the protocol variable decides wheter a rejection will be written to the system-protocol or not.

With the used parameters the following restrictions apply:

- Only ACCEPT, DROP and REJECT can be specified as actions.

- For interface constraints only the output interface can be restricted.

PF_OUTPUT_POLICY This variable describes the default action to be taken if no other rule applies. Possible values:

- ACCEPT
- REJECT
- DROP

PF_OUTPUT_ACCEPT_DEF If this variable is set to 'yes' default rules necessary for correct function of the router will be generated. Use 'yes' as a default here.

If you want to configure the router's behaviour completely yourself you may enter 'no' here but you will have to define all rules on your own then. An equivalent to the default behaviour would look like this:

```
PF_OUTPUT_ACCEPT_DEF='no'

PF_OUTPUT_N='1'
PF_OUTPUT_1='state:ESTABLISHED,RELATED ACCEPT'
```

This single rule accepts only packets belonging to established connections (e.g. packets of the state `ESTABLISHED` or `RELATED`).

PF_OUTPUT_LOG Defines if rejected packets should be logged by the kernel. Log output can be directed to the syslog deamon by activating `OPT_KLOGD`.

PF_OUTPUT_LOG_LIMIT Defines how often log entries will be generated. The frequency is described as *n/time units* with bursts in analog to the limit constraints, e.g. `3/minute:5`. If this entry is empty a default of `1/second:5` is used, if set to `none`, the limit constraints are disabled.

PF_OUTPUT_REJ_LIMIT PF_OUTPUT_UDP_REJ_LIMIT Specifies how often a REJECT-packet is generated when rejecting incoming packets. The frequency is described as *n/time units* with bursts in analog to the limit constraints, e.g. `3/minute:5`. If the limit is exceeded packets will be ignored (DROP). If this entry is empty a default of `1/second:5` is used, if set to `none`, the limit constraints are disabled.

PF_OUTPUT_N PF_OUTPUT_x PF_OUTPUT_x_COMMENT A list of rules that describe which packets the router should transmit resp. drop.

User Defined Lists

In several cases you may want to establish own chains to filter packets in detail there. These chains can be defined and filled with rules via `PF_USR_CHAIN_%`. The names of the chains have to start with *usr-* and after their definition can be used everywhere in the `INPUT`- or `FORWARD`-chain as actions. The ICMP-filter chain used before will serve as an example here:

```
PF_USR_CHAIN_N='1'
#
# create usr-in-icmp
#
PF_USR_CHAIN_1_NAME='usr-in-icmp'
#
# add rule to usr-in-icmp
#
PF_USR_CHAIN_1_RULE_N='2'
PF_USR_CHAIN_1_RULE_1='prot:icmp:echo-request length:0-150 limit:1/second:5 ACCEPT'
PF_USR_CHAIN_1_RULE_2='state:RELATED ACCEPT'
#
# use chain in PF_INPUT
#
PF_INPUT_2='prot:icmp usr-in-icmp'
```

PF_USR_CHAIN_N Defines the number of user defined chains.

PF_USR_CHAIN_x_NAME Defines the name of an user defined chain. The name has to be prefixed by *usr-*.

PF_USR_CHAIN_x_RULE_N

PF_USR_CHAIN_x_RULE_x

PF_USR_CHAIN_x_RULE_x_COMMENT These variables define the rules to be inserted in the user defined chain. All rules may be used that are also valid for the `FORWARD`-chain. If no rule of the user defined chains matches, the router will return to the parent chain and check the next rule after the branching to the user defined rules.

The NAT-Chains (Network Address Translation)

Packets still can be changed after the routing decision. For example they may get a new target address to be forwarded to another computer (port forwarding) or a new source address may be inserted to mask the network behind the router. Masquerading is used i.e. to provide internet access for a private net over one public IP or a in DMZ-setup to hide the structure of the local net from computers in the DMZ.

Configuration is done with two chains, `PREROUTING`- and `POSTROUTING`-chain.
By the `POSTROUTING`-chain the packets are defined that have to be masked by the router. If no rule of the `POSTROUTING`-chain matches, the packets will be forwarded unmasked.

Two variants exist for masquerading: one for network interfaces that do get an IP address allocated on dialin (MASQUERADE) and one for network interfaces with static IP address (SNAT). SNAT in addition expects the source IP address to be inserted into the packet. It may be specified as an:

- IP address (Example: SNAT:1.2.3.4),
- IP range (Example: SNAT:1.2.3.4-1.2.3.10)
- or as symbolic reference (Example: SNAT:IP_NET_1_IPADDR)

For both SNAT and MASQUERADE a port or port range may be set to which the source port may be redirected. Usually this notation is necessary because the kernel can choose the ports on its own. But there exist applications that desire the source port unchanged (and thus require 1:1-NAT) or which forbid PAT (Port Address Translation) or NAPT (Network Address and Port Translation). The port range is simply added to the end, like this: SNAT:IP_NET_1_IPADDR:4000-8000.

With the POSTROUTING-chain only ACCEPT, SNAT, NETMAP and MASQUERADE may be used as actions.

PF_POSTROUTING_N PF_POSTROUTING_x PF_POSTROUTING_x_COMMENT

A list of rules that describe which packets the router should mask resp. forward unmasked. If packets should be excluded from masking an ACCEPT-rule for these packets may be put in front of the MASQUERADE rule.

The PREROUTING-chain configures which packets should be transferred to another computer. If no rule of the PREROUTING-chain matches the packets will be processed further without changes. The action DNAT expects the IP address to be inserted as the target address. It may be specified as an:

- IP address (Example: DNAT:1.2.3.4),
- IP range (Example: DNAT:1.2.3.4-1.2.3.10)
- or as a hostname (Example: DNAT:@client1)

At last a port or port range may be set to which the target port may be redirected. This is only necessary if the target port should be changed. The port (range) is simply added to the end, like this: DNAT:@server:21.

REDIRECT behaves like DNAT, with the exception that the target-IP-address is always set to 127.0.0.1 thus delivering the packet locally. This is needed e.g. for transparent proxies, see OPT_TRANSPROXY (Page 189).

If you want a port forwarded to an interface with a dynamic address you do not know to which IP the packet should be sent (at the time of configuration). Thus you can use dynamic in the PREROUTING-chain as a wildcard for the IP address assigned later on, like this:

```
'dynamic:80  DNAT:1.2.3.4'           # forward http-packets to
                                     # IP address 1.2.3.4
'prot:gre any dynamic DNAT:1.2.3.4'  # forward gre-packets (part of the  PPTP-
                                     # protocol) to IP address 1.2.3.4
```

Only ACCEPT, DNAT, NETMAP and REDIRECT may be used as actions with the PREROUTING-chain.

For further examples on port forwarding see the next paragraph.

PF_PREROUTING_N PF_PREROUTING_x PF_PREROUTING_x_COMMENT

A list of rules that describe which packets should be forwarded to another target by the router.

3.10.5. Example

Below see some examples of the packet filter configuration.

The fli4l Default Configuration

fli4l's default configuration for the INPUT-chain looks like this:

```
PF_INPUT_POLICY='REJECT'
PF_INPUT_ACCEPT_DEF='yes'
PF_INPUT_LOG='no'
PF_INPUT_N='1'
PF_INPUT_1='IP_NET_1 ACCEPT'
```

By this we accomplish that

- computers in the local net are allowed to access the router (PF_INPUT_1='IP_NET_1 ACCEPT'),
- local communication on the router itself is allowed (PF_INPUT_ACCEPT_DEF='yes'),
- packets belonging to connections established by the router are accepted (PF_INPUT_ACCEPT_DEF='yes'),
- everything else is rejected (PF_INPUT_POLICY='REJECT'),
- but nothing is logged to the syslog (PF_INPUT_LOG='no').

The FORWARD-chain looks alike: Only packets of our local net and packets belonging to connections that were established by machines in our local net should be forwarded. In addition NetBIOS- and CIFS-packets will be dropped.

```
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'
PF_FORWARD_N='2'
PF_FORWARD_1='tmpl:samba DROP'
PF_FORWARD_2='IP_NET_1 ACCEPT'
```

Note the dependance on the order of rules: *At first* the NetBIOS-packets are dropped and *afterwards* the packets of the local net are accepted.

The local net may communicate with the router, its packets get forwarded, only the masking which is necessary for the internet access of a local network is still missing:

```
PF_POSTROUTING_N='1'
PF_POSTROUTING_1='IP_NET_1 MASQUERADE'
```

Trusted Nets

If we do want to have several local subnets which should communicate with each other free and unmasked we have to ensure that packets between those nets don't get dropped or masked. In order to achieve this we add a rule or edit the existing one.

Let's assume we have a DSL connection over PPPoE and the two subnets are IP_NET_1 (192.168.6.0/24) and IP_NET_2 (192.168.7.0/24). In this case the configuration would be as follows:

```
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'
PF_FORWARD_N='4'
PF_FORWARD_1='IP_NET_1 IP_NET_2 ACCEPT BIDIRECTIONAL'
PF_FORWARD_2='tmpl:samba DROP'
PF_FORWARD_3='IP_NET_1 ACCEPT'
PF_FORWARD_4='IP_NET_2 ACCEPT'

PF_POSTROUTING_N='3'
PF_POSTROUTING_1='IP_NET_1 IP_NET_2 ACCEPT BIDIRECTIONAL'
PF_POSTROUTING_2='IP_NET_1 MASQUERADE'
PF_POSTROUTING_3='IP_NET_2 MASQUERADE'
```

The first rule ensures forwarding of packets between both subnets without further processing. The third and fourth rule ensure that both subnets also have Internet access. The first rule of the `POSTROUTING`-chain provides unmasked communication between both subnets.

In other words we could say that only packets transferred over the `pppoe`-interface have to be masked:

```
PF_POSTROUTING_N='1'
PF_POSTROUTING_1='if:any:pppoe MASQUERADE'
```

We could as well have restricted the port filtering to the `pppoe`-interface and combined both subnets to one, as seen here:

```
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'
PF_FORWARD_N='2'
PF_FORWARD_1='if:any:pppoe tmpl:samba DROP'
PF_FORWARD_2='192.168.6.0/23 ACCEPT'

PF_POSTROUTING_N='1'
PF_POSTROUTING_1='if:any:pppoe MASQUERADE'
```

Packets going out over the `pppoe`-interface and those addressed to `udp`-ports 137-138 or to `tcp`-ports 139 and 445 will be dropped (rule 1), all other packets from subnet 192.168.6.0/23 will be forwarded (rule 2).

Route Network

Let's add a net 10.0.0.0/24 (i.e. a dial-in network) which we want to communicate with unmasked, but packets to `udp`-ports 137-138 and to `tcp`-Ports 139 and 445 should be dropped:

```
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'
PF_FORWARD_N='4'
PF_FORWARD_1='IP_NET_1 IP_NET_2 ACCEPT BIDIRECTIONAL'
PF_FORWARD_2='tmpl:samba DROP'
PF_FORWARD_3='192.168.6.0/23 ACCEPT'
```

```
PF_FORWARD_4='10.0.0.0/24 ACCEPT'

PF_POSTROUTING_N='2'
PF_POSTROUTING_1='10.0.0.0/24 ACCEPT BIDIRECTIONAL'
PF_POSTROUTING_2='192.168.6.0/23 MASQUERADE'
```

- rule 1 allows unrestricted communication between the subnets `IP_NET_1` and `IP_NET_2`.
- rule 2 drops packets to the samba ports.
- rule 3 and 4 allow forwarding of packets orginating from the subnets 192.168.6.0/24, 192.168.7.0/24 and 10.0.0.0/24; the reverse direction is included by writing `PF_FORWARD_ACCEPT_DEF='yes'`.
- rule 1 of the `POSTROUTING`-chain ensures that packets to resp. from the subnet 10.0.0.0/24-Subnetz are not masked.

An alternative:

```
PF_POSTROUTING_N='1'
PF_POSTROUTING_1='if:any:pppoe MASQUERADE'
```

This rule enables masking only for packets going out over the `pppoe`-interface.

Blacklists, Whitelists

Blacklists (a machine in this list is forbidden to do something) and Whitelists (a machine in this list is allowed to do something) are defined in a very similarl way. Rules are written that are very special at the beginning and to the end are becoming more universal. With a blacklist rules are defined that at the beginning forbid something and at the end allow something to all not previously mentioned. With a Whitelist it is exactly the other way round.

Example 1: All machines in subnet 192.168.6.0/24 except number 12 are allowed to access the Internet as long as they don't use CIFS Ports 137-138 (`udp`), 139 and 445 (`tcp`) to communicate:

```
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'
PF_FORWARD_N='3'
PF_FORWARD_1='192.168.6.12 DROP'
PF_FORWARD_2='tmpl:samba DROP'
PF_FORWARD_3='192.168.6.0/23 ACCEPT'

PF_POSTROUTING_N='1'
PF_POSTROUTING_2='192.168.6.0/24 MASQUERADE'
```

Example 2: Only machine 12 has Internet access (with exception of the ports mentioned above...), all others are only allowed to communicate with another local subnet:

```
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'
PF_FORWARD_N='3'
```

```
    PF_FORWARD_1='192.168.6.0/24 192.168.7.0/24 ACCEPT BIDIRECTIONAL'
    PF_FORWARD_2='tmpl:samba DROP'
    PF_FORWARD_3='192.168.6.12 ACCEPT'

    PF_POSTROUTING_N='1'
    PF_POSTROUTING_1='if:any:pppoe MASQUERADE'
```

3.10.6. Default Configurations

Simple Router Masking A Net Behind Itself

```
#
# Access to the router
#
PF_INPUT_POLICY='REJECT'
PF_INPUT_ACCEPT_DEF='yes'
PF_INPUT_LOG='no'
PF_INPUT_N='1'
PF_INPUT_1='IP_NET_1 ACCEPT'   # all hosts of the local net are allowed
                               # to access the router

#
# Internet access
#
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'

PF_FORWARD_N='2'
PF_FORWARD_1='tmpl:samba DROP' # Samba-packets, that want to leave the
                               # net are dropped
PF_FORWARD_2='IP_NET_1 ACCEPT' # all other packets are allowed
                               # to leave the local net

#
# Maskieren des lokalen Netzes
#
PF_POSTROUTING_N='1'
PF_POSTROUTING_1='IP_NET_1 MASQUERADE'  # mask packets leaving the
                                        # subnet
```

Simple Router Masking Two Nets Behind Itself

```
#
# Access to the router
#
PF_INPUT_POLICY='REJECT'
PF_INPUT_ACCEPT_DEF='yes'
PF_INPUT_LOG='no'
PF_INPUT_N='2'
PF_INPUT_1='IP_NET_1 ACCEPT'   # all hosts of the local net are allowed
                               # to access the router
PF_INPUT_2='IP_NET_2 ACCEPT'   # all hosts of the local net are allowed
```

```
                              # to access the router

#
# Internet access
#
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'

#
# Free communication between the nets
#
PF_FORWARD_N='4'
PF_FORWARD_1='IP_NET_1 IP_NET_2 ACCEPT BIDIRECTIONAL'
PF_FORWARD_2='tmpl:samba DROP' # Samba-packets, that want to leave the
                              # net are dropped
PF_FORWARD_3='IP_NET_1 ACCEPT' # all other packets are allowed
                              # to leave the local net
PF_FORWARD_4='IP_NET_2 ACCEPT' # all other packets are allowed
                              # to leave the local net

#
# Masking of local nets, unmasked communication between those nets
#
PF_POSTROUTING_N='3'
PF_POSTROUTING_1'IP_NET_1 IP_NET_2 ACCEPT BIDIRECTIONAL'
PF_POSTROUTING_2='IP_NET_1 MASQUERADE'  # mask packets leaving the
                                        # subnet
PF_POSTROUTING_3='IP_NET_2 MASQUERADE'  # mask packets leaving the
                                        # subnet
```

Masking DSL-Router With Two Nets Behind It And SSH/HTTP-Access From the Internet

```
#
# Access to the router
#
PF_INPUT_POLICY='REJECT'
PF_INPUT_ACCEPT_DEF='yes'
PF_INPUT_LOG='no'
PF_INPUT_N='4'
PF_INPUT_1='IP_NET_1 ACCEPT'   # all hosts of the local net are allowed
                               # to access the router
PF_INPUT_2='IP_NET_2 ACCEPT'   # all hosts of the local net are allowed
                               # to access the router
PF_INPUT_3='tmpl:ssh ACCEPT'   # allow access to the SSH service
                               # from everywhere
PF_INPUT_4='tmpl:http 1.2.3.4/24 ACCEPT'  # allow machines from
                               # a defined subnet access to the
                               # HTTP service


#
```

```
# Internet access
#
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'

#
# No communication between the nets, both nets have
# Internet access, Samba-packets are dropped
#
PF_FORWARD_N='2'
PF_FORWARD_1='tmpl:samba if:any:pppoe DROP' # Samba-packets, that want to leave the
                              # net are dropped
PF_FORWARD_2='if:any:pppoe ACCEPT' # all other packets are allowed
                              # to leave the local net

#
# Masking of local nets, unmasked communication between those nets
#
PF_POSTROUTING_N='1'
PF_POSTROUTING_1='if:any:pppoe MASQUERADE'  # mask packets leaving the
                                            # subnet
```

Port Forwarding

Port forwarding can be accomplished with the `PREROUTING`-rules like this (`TARGET` refers to the original target address (optional) and the original target port, `NEW_TARGET` refers to the new target address and new target port (optional), `PROTOCOL` refers to the protocol in use):

```
TARGET='<port>'
NEW_TARGET='<ip>'
PROTOCOL='<proto>'
PF_PREROUTING_x='prot:<proto> dynamic:<port> DNAT:<ip>'

TARGET='<port1>-<port2>'
NEW_TARGET='<ip>'
PROTOCOL='<proto>'
PF_PREROUTING_x='prot:<proto> dynamic:<port1>-<port2> DNAT:<ip>'

TARGET='<ip>:<port-a>'
NEW_TARGET='<ip>:<port-b>'
PROTOCOL='<proto>'
PF_PREROUTING_x='prot:<proto> any <ip>:<port-a> DNAT:<ip>:<port-b>'
```

Transparent Proxy

If access to the Internet should only be allowed over a local proxy you may force this behaviour by the help of the `PREROUTING`- and `POSTROUTING`-chains without the client noticing it. In priciple you need to do this in three steps:

1. Redirect all HTTP-port-request to the Proxy except for its own ones (`PREROUTING`).

2. Change the redirected packets in a way that fools the proxy to think they all come from the router so it will return its answers there (`POSTROUTING`).

3. Allow the packets to pass the FORWARD-chain, as far as an entry like

   ```
   PF_FORWARD_x='IP_NET_1 ACCEPT'
   ```

   does not exist (`FORWARD`).

Example 1: Let's assume we only have one net `IP_NET_1`, a squid proxy is running there on a host by the name of `proxy` and the whole `http`-traffic should be processed by it. Squid listens on port 3128. For simplicity we refer via `@proxy` to the host entered in `HOST_1_NAME='proxy'` (see Domain Configuration (Page 66)).

Here are the resulting rules:

```
...
  PF_PREROUTING_x='@proxy ACCEPT'
      # packets from the proxy should not be redirected

  PF_PREROUTING_x='prot:tcp IP_NET_1 80 DNAT:@proxy:3128'
      # HTTP-packets from IP_NET_1 will be redirected to @proxy, Port 3128
      # independet of the target

  PF_POSTROUTING_x='any @proxy:3128 SNAT:IP_NET_1_IPADDR'
      # change all packets to port 3128 in a way as if they came from
      # fli4l (IP_NET_1_IPADDR)

  PF_FORWARD_x='prot:tcp @proxy 80 ACCEPT'
      # let HTTP-packets from the proxy pass the FORWARD-chain (if necessary)
...
```

If more nets or conflicting port forwardings (which are also DNAT-rules) exist, the rules may have to be more differentiated.

Example 2: Our proxy by the name of `proxy` resides in `IP_NET_1`, listens to port 3128 and should only serve clients from `IP_NET_1`. `IP_NET_1` is reachabel over `IP_NET_1_DEV`. Packets from other nets should not be considered.

```
...
  PF_PREROUTING_x='if:IP_NET_1_DEV:any !@proxy 80 DNAT:@proxy:3128'
      # Redirect queries to the HTTP-port that do not emerge from the proxy but
      # come in on an internal interface (IP_NET_1_DEV) to the proxy's port.
      # At this point it is important to check with if:IP_NET_1_DEV:any that the
      # packets are coming from inside because otherwise packets from outside
      # would also be redirected (security breakage)

  PF_POSTROUTING_x='prot:tcp IP_NET_1 @proxy:3128 SNAT:IP_NET_1_IPADDR'
      # Change HTTP-packets originating from IP_NET_1 and destinated to proxy-port 3128
      # in a way as if they came from fli4l (IP_NET_1_IPADDR)

  PF_FORWARD_x='prot:tcp @proxy 80 ACCEPT'
      # let HTTP-packets from the proxy pass the FORWARD-chain (if necessary)
...
```

Example 3: To ease our live and shorten the rules we may use templates (see Using Templates With The Packet Filter (Page 47)). At this point `tmpl:http`, translated in `prot:tcp any any:80` is of advantage. `tmpl:http IP_NET_1 DNAT:@proxy:3128` then changes to `prot:tcp IP_NET_1 80 DNAT:@proxy:3128`.

Both `IP_NET_1` and `IP_NET_2` should be redirected transparently over the proxy. Simplified you could write:

```
...
  PF_PREROUTING_x='tmpl:http @proxy   ACCEPT'
      # HTTP-packets from the proxy should not be redirected

  PF_PREROUTING_x='tmpl:http IP_NET_1 DNAT:@proxy:3128'
      # HTTP-packets from IP_NET_1 should be redirected

  PF_PREROUTING_x='tmpl:http IP_NET_2 DNAT:@proxy:3128'
      # HTTP-packets from IP_NET_2 should be redirected

  PF_POSTROUTING_x='IP_NET_1 @proxy:3128 SNAT:IP_NET_1_IPADDR'
  PF_POSTROUTING_x='IP_NET_2 @proxy:3128 SNAT:IP_NET_2_IPADDR'

  PF_FORWARD_x='tmpl:http @proxy ACCEPT'
...
```

You may continue here forever...

3.10.7. DMZ – Demilitarized Zone

fli4l may also serve to build a DMZ. As this is only another additional ruleset for the router please refer to the wiki at https://ssl.nettworks.org/wiki for the time being.

3.10.8. Conntrack-Helpers

Using IP-Masquerading has the advantage that a bunch of machines in the LAN can be routed over only one official IP-address. However, there are also disadvantages that you have to take into account.

A big problem for example is that no machine from outside can contact the machines in the LAN. This may be desired for security reasons but certain protocols will not work anymore because they require a connection from outside.

A classic example is FTP. Beside a communication channel to exchange commands and answers another channel is needed (an IP-port) to transfer the actual data. fli4l uses certain conntrack-helpers for this in order to open such ports instantaneously and redirect them to the machine in question when needed. The conntrack-helper "listens" to the data stream to recognize when such an additional port is needed.

Typical applications for conntrack-helpers are i.e. chat-protocols and Internet games.

Conntrack-helper are activated over rules in two special arrays. The array `PF_PREROUTING_CT_%` contains helper-assignments to packets coming from outside, the array `PF_OUTPUT_CT_%` contains helper-assignments to packets generated on the router. Some practical examples help to illustrate this.

Example 1: If active FTP from the LAN should be allowed this is, from the router's view, a connection from outside the router, thus an entry in `PF_PREROUTING_CT_%` has to be created:

```
PF_PREROUTING_CT_N='1'
PF_PREROUTING_CT_1='tmpl:ftp IP_NET_1 HELPER:ftp'
```

The `ftp`-helper module will be loaded for all TCP connections from the local network (`IP_NET_1`) to any other addresses' port 21 (which is the `ftp`-Port). This module will allow the FTP server to establish a data transfer connection back to the client during this connection by opening a "hole" in the firewall temporarily.

Example 2: If you want to enable passive ftp for a FTP server on the LAN (the data connection is established from the outside to the inside, so that a hole in the firewall must be opened here as well), this is also seen as a connection from outside by the router. Here we see the rule as for this:

```
PF_PREROUTING_CT_N='1'
PF_PREROUTING_CT_1='tmpl:ftp any dynamic HELPER:ftp'
```

By this rule it is expressed that all FTP connections to the dynamic address of the router are associated to the FTP conntrack helper. Here `dynamic` was used because it is assumed that the router is responsible for dialing in to the Internet and thus has an external IP address. If the router performs dial-in via DSL, the rule can also be written as:

```
PF_PREROUTING_CT_N='1'
PF_PREROUTING_CT_1='tmpl:ftp if:pppoe:any HELPER:ftp'
```

By this rule it is expressed that all FTP connections coming from the DSL interface (`pppoe`) are associated to the conntrack helper.

If the router is not dialing, but e.g. is behind another router (Fritz! box, cable modem, a.s.o.) the following rules can be used:

```
PF_PREROUTING_CT_N='1'
PF_PREROUTING_CT_1='tmpl:ftp if:IP_NET_2_DEV:any HELPER:ftp'
```

It is assumed in the Example, that the connection to the other router is performed over the interface associated with the second subnet (`IP_NET_2_DEV`).

Remember that of course an *additional* configuration of the `FORWARD`-chain is needed to really forward the FTP-packets. A typical rule would be

```
PF_PREROUTING_1='tmpl:ftp any dynamic DNAT:@ftpserver'
```

assuming that the host running the FTP-server has the name `ftpserver`.

Example 3: If you like to use active FTP directly from fli4l (perhaps with the help of the `ftp` program from the `Tools`-package) the firewall has to be prepared, this time in the `OUTPUT`-chain by using the array `PF_output_CT_%`:

```
PF_OUTPUT_CT_N='1'
PF_OUTPUT_CT_1='tmpl:ftp HELPER:ftp'
```

This rule is not necessary if `FTP_PF_ENABLE_ACTIVE='yes'` is used – see the documentation for the `ftp`-OPT in the `tools`-package.

Following is an overview over the existing conntrack-helpers:

Helper	Explanation
ftp	File Transfer Protocol
h323	H.323 (Voice over IP)
irc	Internet Relay Chat
pptp	PPTP Masquerading (By the use of this module it is possible to run more than one PPTP-Client behind the fli4l router at the same time.)
sip	Session Initiation Protocol
sane	SANE Network Procotol
snmp	Simple Network Management Protocol
tftp	Trivial File Transfer Protocol

Table 3.9.: Available Conntrack Helpers In The Packet Filter

Here is an overview over the variables to configure:

PF_PREROUTING_CT_ACCEPT_DEF If this variable is set to 'yes', default rules are generated that are necessary for proper functioning of the router. By default, you should use 'yes' here.

PF_PREROUTING_CT_N PF_PREROUTING_CT_x PF_PREROUTING_CT_x_COMMENT List of rules that describe which incoming packets are associated with conntrack helpers by the router.

PF_OUTPUT_CT_ACCEPT_DEF If this variable is set to 'yes', default rules are generated that are necessary for proper functioning of the router. By default, you should use 'yes' here.

PF_OUTPUT_CT_N PF_OUTPUT_CT_x PF_OUTPUT_CT_x_COMMENT List of rules that describe which packets generated on the router are associated with conntrack helpers by the router.

3.11. Domain configuration

Windows PCs exhibit a somewhat annoying behaviour: If a DNS server is needed and configured at the Windows system, the server is queried regularly (every five minutes) – even if you don't work at the PC!

If you configured an Internet DNS server at your Windows PC, your next bill might become quite expensive :-(

If you don't already run a DNS server in your LAN, this problem can be solved by enabling the DNS server of your fli4l router. The DNS server software used is DNSMASQ.

Before you start configuring your DNS, however, you should give careful consideration to the domain name and the names of the PCs in your network. The domain name you use will not be visible in the Internet. Therefore, you are free to choose any domain name you like.

Additionally, each of your PCs in the LAN has to have a name assigned. These names have to be known by the fli4l router.

DOMAIN_NAME Default Setting: `DOMAIN_NAME='lan.fli4l'`

You can freely choose any domain name as this local domain is not visible in the Internet. However, you should avoid choosing a name that may exist in the Internet (e.g. somewhat.com) because you won't be able to access that Internet domain.

DNS_FORWARDERS Default Setting: `DNS_FORWARDERS=""`

This variable contains the address of your Internet provider's DNS server if you want your fli4l router to route Internet traffic. The fli4l router will forward all DNS queries which it is not able to answer on its own to the address in this variable.

You can specify more than one DNS forwarder by separating the addresses by blanks.

If more DNS server are specified, they will be queried in the order given by the configuration file, meaning the second will only be used in case that the first server does not return a valid answer and so on.

It is also possible to specify a port number for each DNS forwarder address which is then to be separated from the address by a colon. However, in this case it is required to set `OPT_DNS='yes'` (Page 92) (Package dns_dhcp (Page 91)), and you are not allowed to use any of the various `*_USEPEERDNS` options.

Beware: Even if

- `PPPOE_USEPEERDNS` (Page 103),
- `ISDN_CIRC_x_USEPEERDNS` (Page 149) or
- `DHCPCLIENT_x_USEPEERDNS` (Page 91)

are set to 'yes', you need to fill this variable with a valid DNS server address as otherwise no DNS resolution will be possible directly after the router has booted.

Exception: If you use fli4l as a local router *without* a connection to the Internet or other (company) networks with DNS servers, you should set this variable to '127.0.0.1' in order to disable DNS forwarding completely.

HOSTNAME_IP (optional)

This variable can optionally specify to which network 'IP_NET_x' the hostname set by `HOSTNAME` is bound.

HOSTNAME_ALIAS_N (optional)

Number of additional alias host names for the router.

HOSTNAME_ALIAS_x (optional)

Additional alias host name for the router.

3.12. imond configuration

START_IMOND Default setting: `START_IMOND='no'`

`START_IMOND` controls whether to start the imond server or not. The imond server is responsible for monitoring/controlling the fli4l router and for the so-called least cost routing. You can find a detailed description of the Client/Server interface imond (Page 263) in a separate appendix.

Important: The least cost routing funtionality of fli4l can only be used when imond is running. Time-based switching of connections is impossible without imond!

Starting with version 1.5, imond is mandatory for ISDN and DSL routing. In this case you have to set `START_IMOND='yes'`. If you use fli4l as a router between LANs only, you should set `START_IMOND='no'`.

IMOND_PORT The TCP/IP port where imond should wait for connections. You shouldn't change the default value '5000' unless in very exceptional cases.

IMOND_PASS Default setting: `IMOND_PASS=""`

This variable can be used to set a user password for imond. If a client connects to imond at port 5000, imond expects the client to provide this password before processing any requests, with the exception of the commands "quit", "help", and "pass". If you leave `IMOND_PASS` empty, no password is necessary.

The variables

- IMOND_ENABLE (Page 70),
- IMOND_DIAL (Page 69),
- IMOND_ROUTE (Page 69), and
- IMOND_REBOOT (Page 69)

control whether providing the user password is sufficient to execute the control commands like Dial, Hangup, Reboot, or Changing the Default Route, or whether you need a special admin password for these requests (see below).

IMOND_ADMIN_PASS Default setting: `IMOND_ADMIN_PASS=""`

Using the Admin Passwords the client receives all the rights and can thus use all control functions of the server imond – regardless of the content of the variables `IMOND_ENABLE`, `IMOND_DIAL` etc. If you leave `IMOND_ADMIN_PASS` empty, the user password is sufficient to gain all rights!

IMOND_LED The imond server is able to display the router's online/offline state via a LED. This LED is connected to a serial port as follows:

25 pin connector:

```
20 DTR  -------- 1kOhm ----- >| ---------- 7 GND
```

9 pin connector:

```
 4 DTR  -------- 1kOhm ----- >| ---------- 5 GND
```

The LED is on if an ISDN or DSL connection is established, otherwise it is off. If you want this the other way round you have to reverse the polarity of the LED. You can reduce the dropper resistor down to 470 ohm if the LED is lit too dimly.

It is also possible to use two different coloured LEDs. In this case you have to connect the second LED together with a dropper resistor between DTR and GND too, but with

reversed polarity. Then either the first or the second LED will be lit depending on the router's state. Another possibility is to use a DUO LED (two-coloured, three pins).

Currently, the serial port's RTS pin behaves exactly as the DTR pin. You could even attach a third LED for displaying the online/offline state. However, this behaviour may change in the future.

The variable `IMOND_LED` has to be set to the name of the serial port to where the LED is attached; possible values are 'com1', 'com2', 'com3', and 'com4'. Leave the variable empty if you don't use an LED.

IMOND_BEEP If setting `IMOND_BEEP`='yes', imond will emit a two-tone sound over the PC speaker whenever the router's state changes from offline to online and the other way round. In the first case, the higher tone follows the lower one. In the second case, the higher tone is emitted before the lower one.

IMOND_LOG Default setting: `IMOND_LOG='no'`

You can set `IMOND_LOG`='yes' in order to log connections in the file `/var/log/imond.log`. This file can be copied i.e. by scp to another host e.g. for statistical purposes. However, using scp requires you to install and configure the sshd package appropriately.

The structure of the log file entries is described in Table 3.10.

Table 3.10.: Structure of Imond log files

Entry	Meaning
Circuit	the name of the circuit for which the entry has been created
Start time	the date and time of dialing this circuit
Stop time	the date and time of hanging up this circuit
Online time	the time this circuit was online
Billed time	the time for which the provider will charge you (depends on the timing)
Costs	the costs the provider will charge to your account
Bandwidth	the bandwidth used, separated into "in" and "out" ("in" coming first), presented as two unsigned integer numbers for which the following applies: Bandwidth = *4GiB* $*$ *<first number>* + *<second number>*
Device	the device used for communication
Invoice pulse	the invoice pulse used by the provider for charging (taken from the circuit configuration)
Call charge	the fee charged per invoice pulse (taken from the circuit configuration)

The costs are denoted in Euro. These values are only meaningful if you correctly define the corresponding circuit variables `ISDN_CIRC_x_TIMES` (Page 155).

IMOND_LOGDIR If the imond log is activated, this variable can be used to choose an alternative log directory instead of the default /var/log, e.g. /boot. This is useful in order to make the log persistent on the boot medium. However, this requires the boot medium to be mounted read/write.

The default value is 'auto' which lets the fli4l router to determine the storage location automatically. Depending on further configuration, the storage path is /boot/persistent/base or some other path determined by the FLI4L_UUID variable. If neither FLI4L_UUID is set nor /boot is mounted read/write, the log file can be found under /var/run.

IMOND_ENABLE IMOND_DIAL IMOND_ROUTE IMOND_REBOOT These variables make certain imond commands available in user mode (enabling/disabling the ISDN interface, dialing/hanging up, changing the default route, rebooting the router).

Default settings:

```
IMOND_ENABLE='yes'
IMOND_DIAL='yes'
IMOND_ROUTE='yes'
IMOND_REBOOT='yes'
```

All other features of imond's Client-Server interface are described in a separate chapter (Page 263).

3.13. General circuit configuration

IP_DYN_ADDR If you use connections with dynamic IP address assignment, you need to set `IP_DYN_ADDR` to 'yes', otherwise to 'no'. Most Internet providers use dynamic IP address assignment.

Default setting: `IP_DYN_ADDR='yes'`

DIALMODE fli4l's default dial mode is 'auto', i.e. fli4l dials automatically if an IP packet has to be routed to an IP address outside the LAN. However, you may also set the dial mode to 'manual' or 'off'. In these cases, dialing to establish a connection is only possible using the imonc client or the Web-Interface.

Default setting: `DIALMODE='auto'`

4. Packages

Besides the BASE installation there are also packages. Each package contains one or more "OPTs" [1] which can be installed in addition to the base installation. Some of the OPTs are part of the BASE package, other have to be downloaded separately. The download site (http://www.fli4l.de/en/download/stable-version/) gives an overview over the packages provided by the fli4l team, the OPT database (http://extern.fli4l.de/fli4l_opt-db3/) contains packages offered by other authors. In the following, we describe the packages supplied by the fli4l team.

4.1. Tools In The Package 'Base'

The following OPTs are contained in the BASE package:

Name	Description
OPT_SYSLOGD	Tool for logging system messages (Page 71)
OPT_KLOGD	Tool for logging kernel messages (Page 73)
OPT_LOGIP	Tool for logging WAN IP addresses (Page 73)
OPT_Y2K	Date correction utility for systems that are not Y2K-safe (Page 73)
OPT_PNP	Installation of ISAPnP tools (Page 74)

4.1.1. OPT_SYSLOGD – Logging system messages

Many programs use the Syslog interface to log messages. If you want to see these messages on your fli4l console you have to start the syslogd daemon.

Setting OPT_SYSLOGD to 'yes' enables debugging messages, 'no' disables them.

See also ISDN_CIRC_x_DEBUG (Page 154) and PPPOE_DEBUG (Page 103).

Default Setting: OPT_SYSLOGD='no'

SYSLOGD_RECEIVER SYSLOGD_RECEIVER controls whether fli4l can receive Syslog messages from other hosts in the network.

SYSLOGD_DEST_N SYSLOGD_DEST_x SYSLOGD_DEST_x describes where the system messages being received by syslogd should be displayed. Normally, this is fli4l's console, hence:

```
SYSLOGD_DEST_1='*.* /dev/console'
```

If you want to log the messages into a file, you can e.g. use:

```
SYSLOGD_DEST_1='*.* /var/log/messages'
```

[1] abbreviation for "OPTional module"

If you have a so-called “log host” in your network you can redirect the Syslog messages to that host if you supply its IP address.

Beispiel:

```
SYSLOGD_DEST_1='*.* @192.168.4.1'
```

The “@” sign has to be prepended to the IP address.

If you want the Syslog messages to be delivered to multiple destinations it is necessary to increase the variable `SYSLOGD_DEST_N` (number of destinations used) accordingly and to fill the variables `SYSLOG_DEST_1`, `SYSLOG_DEST_2` etc. with appropriate content.

The syntax ‘*.*’ directs syslogd to log all messages. However, you are also able to constrain the messages to be logged for certain destinations by the use of so-called “priorities”. In this case you need to replace the asterisk (*) after the dot (.) by one of the following keywords:

- debug
- info
- notice
- warning (deprecated: warn)
- err (deprecated: error)
- crit
- alert
- emerg (deprecated: panic)

The items in the list are descending sorted according to severity. The keywords “error”, “warn”, and “panic” are deprecated—you should not use them anymore.

You can replace the asterisk (*) in front of the dot by a so-called “facility”. However, a detailed explanation is outside this scope. You can find an overview over the available facilities at the man page of syslog.conf:

http://linux.die.net/man/5/syslog.conf

In most cases an asterisk is completely sufficient. Example:

```
SYSLOGD_DEST_1='*.warning @192.168.4.1'
```

Windows hosts can serve as log hosts as well as Unix/Linux hosts. You can find links to adequate software at http://www.fli4l.de/en/other/links/. Using a log host is strongly recommended if you want a detailed logging protocol. The protocol is also useful for debugging purposes. The Windows client imonc also “understands” the Syslog protocol and is able to display the messages in a window.

Unfortunately, messages generated during the boot process cannot be directed to syslogd. However, you can configure fli4l to use a serial port as a terminal. You can find more information on this topic in the section Console settings (Page 29).

SYSLOGD_ROTATE You can use `SYSLOGD_ROTATE` in order to control whether Syslog message files are rotated once a day, thereby archiving the messages of the last x days.

SYSLOGD_ROTATE_DIR The optional variable `SYSLOGD_ROTATE_DIR` lets you specify the directory where the archived Syslog files should be stored. Leave it empty to use the default directory /var/log.

SYSLOGD_ROTATE_MAX The optional variable `SYSLOGD_ROTATE_MAX` lets you specify the number of archived/rotated Syslog files.

SYSLOGD_ROTATE_AT_SHUTDOWN With the optional variable `SYSLOGD_ROTATE_AT_SHUTDOWN` you can disable the rotate of syslog files at shutdown. Please only do this, if your syslogfiles are written directly to a destination on a permanent disk.

4.1.2. OPT_KLOGD – Logging kernel messages

Many errors, e.g. a dial-in that failed, are written directly to the console by the Linux kernel. If you set `OPT_KLOGD`='yes', these messages are redirected to the Syslog daemon which can log them to a file or send them to a log host (see above). This keeps your fli4l console (almost) clear.

Recommendation: If you use `OPT_SYSLOGD`='yes' you should also set `OPT_KLOGD` to 'yes'.

Default setting: `OPT_KLOGD`='no'

4.1.3. OPT_LOGIP – Logging WAN IP addresses

LOGIP logs your WAN IP address to a log file. You activate this logging by setting `OPT_LOGIP` to 'yes'.

Default setting: `OPT_LOGIP`='no'

LOGIP_LOGDIR – Configure directory of log file

The variable `LOGIP_LOGDIR` contains the directory where the log file should be created or 'auto' for autodetect.

Default setting: `LOGIP_LOGDIR`='auto'

4.1.4. OPT_Y2K – Date correction for systems that are not Y2K-safe

fli4l routers are often assembled from old hardware parts. Older mainboards may have a BIOS that is not Y2K-safe. This can lead to the situation that setting the system date to the 27th May 2000 causes the BIOS date to become the 27th May 2094 after a reboot. By the way, Linux will then show the 27th May 1994 as system date.

Normally the system date reflected by fli4l is not important and should not matter at all. If you use the LCR (Least Cost Routing) functionality of your fli4l router this may very well play a role.

The reason: The 27th May 1994 was a Friday, the 27th May 2000 in contrast was a Saturday. And for the weekend there are lower-priced rates or providers, respectively ...

A first solution to that problem is as follows: The BIOS date is changed from the 27th May 2000 to the 28th May 1994 which was a Saturday, too. However, the problem is not solved completely yet: Not only does fli4l use the day of week and the current time for least-cost routing but it also respects bank holidays.

Y2K_DAYS – add N days to the system date

Because the BIOS date differs from the actual one by exactly 2191 days, the setting

```
Y2K_DAYS='2191'
```

causes the fli4l router to add 2191 days to the BIOS date before using it as the Linux system date The BIOS date is left untouched because otherwise the year would be wrong (2094 or 1994, resp.) again after the next boot.

There is an additional alternative:

Using a time server, fli4l is able to fetch the current date and time from the Internet. The package CHRONY (Page **??**) is designed for this purpose. Both settings can be combined. This is useful as it allows to correct the date via Y2K_DAYS before setting the exact time using the information from the time server.

If you do not have any problems with Y2K, set OPT_Y2K='no' and forget it ...

4.1.5. OPT_PNP – Installation of ISAPnP tools

Some ISAPnP adapters have to be configured by the "isapnp" tool. This especially affects ISDN adapters with a ISDN_TYPE of 7, 12, 19, 24, 27, 28, 30, and 106 – but only if the adapter is really an ISAPnP adapter.

For proper configuration you have to create the file "etc/isapnp.conf".

Brief instructions to create this file follow:

- In <config>/base.txt, set OPT_PNP='yes' and MOUNT_BOOT='rw'
- boot your fli4l – the ISAPnP adapter will most likely not be detected
- Logon to the fli4l's console and type:

  ```
  pnpdump -c >/boot/isapnp.conf
  umount /boot
  ```

  This saves the ISAPnP configuration to your boot medium.

Continue on your PC (Unix/Linux/Windows):

- Copy the file isapnp.conf from your boot medium to <config>/etc/isapnp.conf
- Edit isapnp.conf and save your changes
  The default values can be left unchanged or be replaced by the values proposed. The relevant lines are shown in the following example:

```
     #     Start dependent functions: priority acceptable
     #       Logical device decodes 16 bit IO address lines
     #             Minimum IO base address 0x0160
     #             Maximum IO base address 0x0360
     #             IO base alignment 8 bytes
     #             Number of IO addresses required: 8
1)      (IO 0 (SIZE 8) (BASE 0x0160))
     #       IRQ 3, 4, 5, 7, 10, 11, 12 or 15.
     #             High true, edge sensitive interrupt (by default)
2)      (INT 0 (IRQ 10 (MODE +E
```

1) – Here, you can choose the I/O „BASE“ address. This address must lie between the minimum and maximum address and conform to the „base alignment“.
If your system uses more than one ISA adapter, you will have to ensure that there are no overlaps between address ranges. The address range starts at „BASE“ and ends at „BASE + Number of IO addresses required“.

2) – Here you can pick an IRQ from the list shown. The IRQs 2(9), 3, 4, 5, and 7 should be avoided as these IRQs may clash with your serial and parallel interfaces or the interrupt cascading.
ISA adapters are not able to share IRQs, thus the IRQ you choose here may not be used elsewhere.

- Put the chosen configuration (IRQ/IO) into <config>/isdn.txt
- In order to let the necessary files be copied to the boot medium, you must set `OPT_PNP` to 'yes' in <config>/base.txt. The variable `MOUNT_BOOT` can be chosen freely, however.
- Create new boot medium

The automatically generated file contains Unix line endings (LF without CR). Thus, if you use Notepad under Windows, all content is shown in a single line. In contrast, the DOS editor "edit" is able to cope with the Unix line endings. When saved, however, they are changed to DOS line endings (CR+LF).

Workaround:

- start DOS box
- change to the directory <config>/etc
- type: edit isapnp.conf
- edit file and save your changes

After that you can also use Notepad to edit the file.

Under Windows you may also use the Wordpad editor.

The CRs generated by the "edit" tool are filtered when fli4l boots and thus do not disturb.

Please try first to get along without using `OPT_PNP`. If the adapter is not recognized you may follow the procedure described above.

If you update to a more recent fli4l version, you may reuse the previously created isapnp.conf.

Default setting: `OPT_PNP`='no'

4.2. Advanced Networking

The package 'advanced networking' provides bonding and bridging capabilities for the fli4l-router. EBTables (http://ebtables.sourceforge.net/) support can be enabled as well. This allows to build a transparent packet filter. To all options of the advanced_networking package generally applies:

This package is only for users with profund knowledge about networks and routing.

Very unusual problems can appear especially using EBTables without perfectly knowing the diverse operational modes of layer 2 and 3. Some filtering rules of the packet filter will work completely different with EBTables support enabled.

4.2.1. Broadcast Relay - Forwarding of IP Broadcasts

Using a Broadcast Relay, IP broadcasts can surpass interface boundaries. This is necessary for applications which determine network devices using broadcasts (eg QNAP Finder). Broadcasts are normally not passed across network boundaries by the router. This problem can be circumvented by using a broadcast relay.

Within a Broadcast Relay broadcasts are always forwarded to all connected interfaces. This means that setting up a second Broadcast Relay with interfaces swapped is not necessary. In addition, multiple broadcast relays including the same interface are not allowed.

OPT_BCRELAY Broadcast Forwarding

Default: `OPT_BCRELAY='no'`

Setting `'yes'` here activates the Broadcast Relay package. Specifying `'no'` deactivates the Broadcast Relay package completely.

BCRELAY_N Default: `BCRELAY_N='0'`

The number of Broadcast Relays to configure.

BCRELAY_x_IF_N Default: `BCRELAY_x_IF_N='1'`

Number of interfaces assigned to this broadcast relay.

BCRELAY_x_IF_x Default: `BCRELAY_x_IF_x="` Name of the interface assigned to this broadcast relay.

For illustration an example follows where the computer with the application (eg QNAP Finder) is located in the internal network (connected to `eth0`) and the NAS is in a different network (connected to `eth1`).

```
OPT_BCRELAY='yes'
BCRELAY_N='1'
BCRELAY_1_IF_N='2'
BCRELAY_1_IF_1='eth0'
BCRELAY_1_IF_2='eth1'
```

4.2.2. Bonding - Combining Several Network Interface Cards In One Link

Bonding refers to joining at least two network interface cards into one link. The cards even may be of different type (ie 3Com and Intel) or speed (ie 10 Mbit/s or 100 Mbit/s). You can either connect linux computers directly or connect to a network switch using bonding. In this way a 200 Mbit/s full duplex connection from a flil4-router to a switch can be used without much effort. Everyone interested in using bonding should have read the documentation in the kernel directory (bonding.txt). The names of the bonding settings largely correspond to the names used there.

OPT_BONDING_DEV Default: `OPT_BONDING_DEV='no'`

`'yes'` activates the bonding package, `'no'` deativates the bonding package completely.

BONDING_DEV_N Default: `BONDING_DEV_N='0'`

Number of bonding devices to be configured.

BONDING_DEV_x_DEVNAME Default: `BONDING_DEV_x_DEVNAME=''`

Name of the bonding device to be created. It should consist of the prefix 'bond' and a trailing number with out a leading '0'. The numbers of the bonding devices don't have to start with '0' and need not be consecutive. Possible values could be 'bond0', 'bond8' or 'bond99'.

BONDING_DEV_x_MODE Default: `BONDING_DEV_x_MODE=''`

Specifies the bonding method. Default is round-robin 'balance-rr'. Possible values are listed below:

balance-rr Round-robin method: Submit sequentially over all slaves from the first to the last. This method provides both load balancing and fault tolerance.

active-backup Active backup: Only one slave in the bond is active. The other slaves are activated only when the active slave fails. The MAC address of the bond is only visible on one port (network adapter) so it does not confuse the switch. This mode provides fault tolerance.

balance-xor XOR method: Submit based on the formula [(Source-MAC-address XOR destination-MAC-address) modulo the number of slaves]. This ensures that the same slave always is used for the same destination-MAC-address. This method provides both load balancing and fault tolerance.

broadcast Broadcast method: Transmits everything on all slave devices. This mode provides fault tolerance.

802.3ad IEEE 802.3ad dynamic link aggregation. Creates aggregation groups that share the same speed and duplex settings. Transmits on all slaves in the active aggregator.

Requirements:

- ethtool support in the base device driver to retrieve speed and duplex status for each device.
- a switch that supports dynamic IEEE 802.3ad connection aggregation.

balance-tlb Adaptive load balancing for outgoing data: channel bonding that does not need any special features in the switch. The outgoing network traffic is distributed on each slave according to the current load. Incoming network traffic is received by the current slave. If the receiving slave fails, another slave takes over the MAC address of the slave gone down.

Requirements:

- ethtool support in the base device driver to retrieve speed and duplex status for each device.

balance-alb Adaptive load balancing: includes both balance-tlb, and inbound load balancing (rlb) for IPV4 traffic and needs no special requirements on the Switch. Load Balancing for incoming traffic is achieved through ARP requests. The bonding driver catches ARP responses from the server on their way outside and overrides the source hardware address with the unique hardware address of a slave in the bond. This way different clients use different hardware addresses for the server.

Incoming traffic from connections created by the server will also be balanced. If the server sends ARP requests, the bonding driver copies and stores the client IP from the ARP. At the time the ARP response of the client arrives the bonding driver determines its hardware address and creates an ARP reply to this client assigning a client in the bond to it. A problematic effect of ARP arrangements for load balancing is that every time an ARP request is sent the hardware address of the bond is used. Clients learn the hardware address of the bond and the incoming traffic on the current slave collapses. This fact is countered in a way that updates (ARP Replies) to all clients will be sent to their respective hardware addresses so that the traffic is divided again. Incoming traffic will be newly allocated even when a new slave is added to the bonding or an inactive slave is re-activated. The receiving load is distributed sequentially (round robin) in the group of the slave with the largest network speed in the bond.

When a connection is restored or a new slave joins the bond incoming traffic will be distributed anew to all active Slaves in the bond by sending ARP replies with the selected MAC addresses to each client. The parameter 'updelay' must be set to a value greater than or equal to the forwarding delay of the switch in order to avoid blocking of ARP responses to clients.

Requirements:

- ethtool support in the base device driver to retrieve speed and duplex status for each device.
- support in the base device driver to set the hardware address even when the device is open. This is neccessary for granting that at every time at least one slave in the bond is carrying the hardware address of the bond (curr_active_slave) although every slave in the bond has its own unique hardware address. If curr_active_slave fails its hardware address will simply be replaced with a new one.

BONDING_DEV_x_DEV_N Default: `BONDING_DEV_x_DEV_N='0'`

Specifies the number of physical devices the bond consists of. E.g. for a bond between 'eth0' and 'eth1' (two eth-devices) '2' has to be entered.

BONDING_DEV_x_DEV_x Default: `BONDING_DEV_x_DEV_x="`

The name of a physical device which belongs to this bonding device. An example would be the value 'eth0'. Please note that a physical device that you use for a bond can't be used for anything else. So you can't use it additionally for a DSL modem, a bridge, a VLAN or inclusion in base.txt.

BONDING_DEV_x_MAC Default: `BONDING_DEV_x_MAC="`

This setting is optional and can also be completely omitted.

A bonding device defaults to the MAC address of the first physical device which is used for bonding. If you do not want this it is possible to specify a MAC address the bonding device should use here.

BONDING_DEV_x_MIIMON Default: `BONDING_DEV_x_MIIMON='100'`

This setting is optional and can also be completely omitted.

Specifies the interval (in milliseconds) in which the individual connections of a bonding device are checked for their link status. The link status of each physical device in the bond will be checked every x milliseconds. Setting this to '0' will disable the miimon monitoring.

BONDING_DEV_x_USE_CARRIER Default: `BONDING_DEV_x_USE_CARRIER='yes'`

This setting is optional and can also be completely omitted.

If the link status check by miimon (see above) is enabled this setting can specify the function performing the check.

- `'yes'`: netif _carrier_ok() function
- `'no`: direct calls to MII or ethtool ioctl() System calls

The netif_carrier_ok() method is more efficient, but not all drivers do support this method.

BONDING_DEV_x_UPDELAY Default: `BONDING_DEV_x_UPDELAY='0'`

This setting is optional and can also be completely omitted.

The value of this setting multiplied by the setting of `BONDING_DEV_x_MIIMON` specifies the time a in which a connection of bonding devices is activated after the corresponding link (for example 'eth0') is up. This way the connection of the bonding device is activated until the link status switches to "not connected".

BONDING_DEV_x_DOWNDELAY Default: `BONDING_DEV_x_DOWNDELAY='0'`

This setting is optional and can also be completely omitted.

The value of this setting multiplied by the setting of `BONDING_DEV_x_MIIMON` specifies the time a in which a connection of bonding devices is deactivated if the appropriate link (iE an eth-device) fails. This will deactivate the connection of a bonding device temporarily until the link status is back to 'active'.

BONDING_DEV_x_LACP_RATE Default: `BONDING_DEV_x_LACP_RATE='slow'`

This setting is optional and can also be completely omitted.

Specifiy how often link informations are exchanged between the link partners (for example a switch or another linux PC) if `BONDING_DEV_x_MODE="` is set to '802.3ad'.

- 'slow': every 30 seconds
- 'fast': each second.

BONDING_DEV_x_PRIMARY Default: `BONDING_DEV_x_PRIMARY="`

This setting is optional and can also be completely omitted.

Specify primary output device if mode is set to 'active-backup'. This is useful if the various devices have different speeds. Provide a string (for example 'eth0') for the device to be used primarily. If a value is entered and the device is online it will be used as the first output medium. Only if the device is offline another device will be used. If a failure is detected a new standard output medium will be chosen. This comes in handy if one slave has priority over another, for example if a slave is faster than another (1000 Mbit/s versus 100 Mbit/s). If the 1000 Mbit/s slave fails and later gets back up it may be of advantage to set the faster slave active again without having to cause a fail of the 100 Mbit/s slave artificially (for example by pulling the plug).

BONDING_DEV_x_ARP_INTERVAL Default: `BONDING_DEV_x_ARP_INTERVAL='0'`

This setting is optional and can also be completely omitted.

The interval in which IP-addresses specified in `BONDING_DEV_x_ARP_IP_TARGET_x` are checked by using their ARP responses (in milliseconds). If ARP monitoring is used in load-balancing mode (mode 0 or 2) the switch should be adjusted to distribute packets to all connections equally (for example round robin). If the switch is set to distribute the packets according to the XOR method all responses of the ARP targets will arrive on the same connection which could cause failure for all team members. ARP monitoring should not be combined with miimon. Passing '0' will disable ARP monitoring.

BONDING_DEV_x_ARP_IP_TARGET_N Default: `BONDING_DEV_x_ARP_IP_TARGET_N="`

This setting is optional and can also be completely omitted.

The number of IP-addresses which are used for ARP checking. A maximum of 16 IP-addresses can be checked.

BONDING_DEV_x_ARP_IP_TARGET_x Default: `BONDING_DEV_x_ARP_IP_TARGET_x="`

This setting is optional and can also be completely omitted.

If `BONDING_DEV_x_ARP_INTERVAL` is > 0, specify one IP address which is used as the target for ARP requests to evaluate the quality of the connection. Enter values using format 'ddd.ddd.ddd.ddd'. To get ARP monitoring to work at least one IP address has to be given here.

4.2.3. VLAN - 802.1Q Support

Support for 802.1Q VLAN is reasonable only in conjunction with using appropriate switches. Port-based VLAN switches are *not* suitable. A general introduction to the subject VLAN can be found at http://en.wikipedia.org/wiki/IEEE_802.1Q. At http://de.wikipedia.org/wiki/VLAN some additional information can be found.

Please note that not any network card can handle VLANs. Some can not handle VLANs at all, others require a matching MTU and few cards work without any problems. The author of the advanced_networking package uses Intel network cards with the 'e100' driver without any problem. MTU adjustment is not necessary. 3COM's '3c59x' driver requires MTU adjustment to 1496 otherwise the card won't work correctly. The 'starfire' driver does not work properly if a VLAN device is added to a bridge. In this case no packets can be received. Those who want

to work with VLANs should ensure that the respective Linux NIC drivers support VLANs correctly.

OPT_VLAN_DEV Default: `OPT_VLAN_DEV='no'`

`'yes'` activates the VLAN package, `'no'` deactivates it.

VLAN_DEV_N Default: `VLAN_DEV_N="`

Number of VLAN devices to configure.

VLAN_DEV_x_DEV Default: `VLAN_DEV_x_DEV="`

Name of the device connected to a VLAN capable switch (iE `'eth0'`, `'br1'`, `'eth2'`...).

VLAN_DEV_x_VID Default: `VLAN_DEV_x_VID="`

The VLAN ID for which the appropriate VLAN device should be created. The name of the VLAN device consists of the prefix 'ethX' and the attached VLAN ID (without leading '0'). For example '42' creates a VLAN device `'eth0.42'` on the fli4l-router.

VLAN devices on the fli4l-router are always named '<device>.<vid>'. So if you have an eth-device connected to a VLAN-capable switch and you want to use VLANs 10, 11 and 23 on the fli4l-router you have to configure 3 VLAN devices with the eth-device as `VLAN_DEV_x_DEV='ethX'` and the respective VLAN ID in `VLAN_DEV_x_VID="`. Example:

```
OPT_VLAN_DEV='yes'
VLAN_DEV_N='3'
VLAN_DEV_1_DEV='eth0'
VLAN_DEV_1_VID='10' # will create device: eth0.10
VLAN_DEV_2_DEV='eth0'
VLAN_DEV_2_VID='11' # will create device: eth0.11
VLAN_DEV_3_DEV='eth0'
VLAN_DEV_3_VID='23' # will create device: eth0.23
```

Please always remember to check the MTU of all units involved. Caused by the VLAN header the frames will be 4 bytes longer. If necessary the MTU must be changed to 1496 on the devices.

4.2.4. Device MTU - Adjusting MTU Values

In rare circumstances it may be necessary to adjust the MTU of a device. E.G. some not 100% VLAN-compatible network cards need to adjust the MTU. Please remember that few network cards are capable of processing Ethernet frames larger than the 1500 bytes!

DEV_MTU_N Default: `DEV_MTU_N="`

This setting is optional and can also be completely omitted.

Number of devices to change their MTU settings.

DEV_MTU_x Default: `DEV_MTU_x="`

This setting is optional and can also be completely omitted.

Name of the device to change its MTU followed by the MTU to be set. Both statements have to be separated by a space. To set a MTU of `'1496'` for `'eth0'` enter the following:

```
DEV_MTU_N='1'
DEV_MTU_1='eth0 1496'
```

4.2.5. BRIDGE - Ethernet Bridging for fli4l

This is a full-fledged ethernet-bridge using spanning tree protocol on demand. For the user the Computer seems to work as a layer 3 switch on configured ports.
Further information on bridging can be found here:

- Homepage of the Linux Bridging Project:
  http://bridge.sourceforge.net/

- The detailed and authoritative description of the bridging standards:
  http://standards.ieee.org/getieee802/download/802.1D-2004.pdf.
  (Mainly informations from page 153 on are interesting. Please note that the Linux bridging code is working according to standards from 1998, allowing only 16 bit Values for pathcost as an example.)

- Calculation of different timing values for the spanning tree protocol:
  http://www.dista.de/netstpclc.htm

- See how STP is working by looking at some nice examples:
  http://web.archive.org/web/20060114052801/http://www.zyxel.com/support/supportnote/ves1012/app/stp.htm

OPT_BRIDGE_DEV Default: `OPT_BRIDGE_DEV='no'`

`'yes'` activates the bridge package, `'no'` deactivates it.

BRIDGE_DEV_BOOTDELAY Default: `BRIDGE_DEV_BOOTDELAY='yes'`

This setting is optional and can also be completely omitted.

As a bridge needs at least 2 × `BRIDGE_DEV_x_FORWARD_DELAY` in seconds to become active this period has to be waited if devices are needed at the startup of the fli4l-router. As an example consider sending syslog messages or dialing in via DSL. If the entry is set to `'yes'` 2 × `BRIDGE_DEV_x_FORWARD_DELAY` is waited automatically. If the bridges are not required at startup-time `'no'` should be set to accelerate the startup process of fli4l router.

BRIDGE_DEV_N Default: `BRIDGE_DEV_N='1'`

The number of independent bridges. Each bridge has to be considered completely isolated. This applies in particular for the setting of `BRIDGE_DEV_x_STP`. There will be created one virtual device by the name of `'br<nummer>'` per bridge.

BRIDGE_DEV_x_NAME Default: `BRIDGE_DEV_x_NAME="`

The symbolic name of the bridge. This name can be used by other packages in order to use the bridge regardless of its device name.

BRIDGE_DEV_x_DEVNAME Default: `BRIDGE_DEV_x_DEVNAME="`

Each bridge device needs a name in the form of `'br<number>'`. <number> can be a number between '0' and '99' without leading '0'. Possible entries could be `'br0'`, `'br9'` or `'br42'`. Names can be chosen arbitrary, the first bridge may be `'br3'` and the second `'br0'`.

BRIDGE_DEV_x_DEV_N Default: `BRIDGE_DEV_x_DEV_N='0'`

How many network devices belong to the bridge? The count of devices that should be connected to the bridge. It can even be `'0'` if the bridge is only a placeholder for an IP-address that should be taken over by a VPN-tunnel connected to the bridge.

BRIDGE_DEV_x_DEV_x_DEV Specifies which device can be connected to the bridge. You can fill in an eth-device (ie `'eth0'`), a bonding device (iE `'bond0'`) or also a VLAN-device (iE `'vlan11'`). A device connected here may not be used in other places and is not allowed to get an IP-address assigned.

```
BRIDGE_DEV_1_DEV_N='3'
BRIDGE_DEV_1_DEV_1_DEV='eth0.11' #VLAN 11 on eth0
BRIDGE_DEV_1_DEV_2_DEV='eth2'
BRIDGE_DEV_1_DEV_3_DEV='bond0'
```

BRIDGE_DEV_x_AGING Default: `BRIDGE_DEV_x_AGING='300'`

This setting is optional and can also be completely omitted.

Specifies the time after which old entries in the bridges' MAC table will be deleted. If in this amount of time in seconds no data is received or transmitted by the computer with the network card the corresponding MAC address will be deleted in the bridges' MAC table.

BRIDGE_DEV_x_GARBAGE_COLLECTION_INTERVAL

Default: `BRIDGE_DEV_x_GARBAGE_COLLECTION_INTERVAL='4'`

This setting is optional and can also be completely omitted.

Specifies the time after which „garbage collection“ will be done. All dynamic entries will be checked. Entries not longer valid and outdated will get deleted. In particular old invalid connections will be deleted.

BRIDGE_DEV_x_STP Default: `BRIDGE_DEV_x_STP='no'`

This setting is optional and can also be completely omitted.

Spanning tree protocol allows to manage multiple connections to different switches. This results in redundancy ensuring network functionality in case of line failures. Without the use of STP redundant lines between switches aren't possible and networking may fail. STP tries to use the fastest connection between two switches. This way even connections with different speeds are reasonable. You may iE use a 1 Gbit/s connection as main and a second 100 Mbit/s as a fallback.

A good source of background informations can be found here:
http://en.wikipedia.org/wiki/Spanning_Tree_Protocol.

BRIDGE_DEV_x_PRIORITY Default: `BRIDGE_DEV_x_PRIORITY="`

This setting is optional and can also be completely omitted.

Only valid if `BRIDGE_DEV_x_STP='yes'` is set!

Which priority has this bridge? The bridge with the lowest priority wins the main bridge election. Each bridge should have a different priority. Please note that the bridge with the lowest priority should also have the biggest available bandwith because in addition to the complete data traffic control packets will be sent by it every 2 seconds. (See also: `BRIDGE_DEV_x_HELLO`)

Valid Values are from `'0'` to `'61440'` in steps of 4096.

BRIDGE_DEV_x_FORWARD_DELAY Default: `BRIDGE_DEV_x_FORWARD_DELAY='15'`

This setting is optional and can also be completely omitted.

Only valid if `BRIDGE_DEV_x_STP='yes'` is set!

If one connection of the bridge was deactivated or if a connection is added to the bridge it takes the given time in seconds × 2 until the connection can send data. This parameter is crucial for the time the bridge needs to recognize a dead connection. The time period is calculated in seconds with this formula:

`BRIDGE_DEV_x_MAX_MESSAGE_AGE`+(2 × `BRIDGE_DEV_x_FORWARD_DELAY`)

In standard values this means: $20 + (2 \times 15) = 50$ seconds. The time to recognize a dead connection can be minimized if `BRIDGE_DEV_x_HELLO` is set to 1 second and `BRIDGE_DEV_x_FORWARD_DELAY` is set to 4 seconds. In addition `BRIDGE_DEV_x_MAX_MESSAGE_AGE` has to set to 4 seconds. This leads to: $4 + (2 \times 4) = 12$ seconds. This is as fast as it can get.

BRIDGE_DEV_x_HELLO Default: `BRIDGE_DEV_x_HELLO='2'`

This setting is optional and can also be completely omitted.

Only valid if `BRIDGE_DEV_x_STP='yes'` is set!

The time mentioned in `BRIDGE_DEV_x_HELLO` is the time in seconds in which the so-called 'Hello-message' is sent by the main bridge. These messages are necessary for STP's automatic configuration.

BRIDGE_DEV_x_MAX_MESSAGE_AGE Default: `BRIDGE_DEV_x_MAX_MESSAGE_AGE='20'`

This setting is optional and can also be completely omitted.

Only valid if `BRIDGE_DEV_x_STP='yes'` is set!

The maximum time period the last 'Hello-message' stays valid. If no new 'Hello-message' is received during this period a new main bridge election will be triggered. This is why this value should **never** be lower than 2 × `BRIDGE_DEV_x_HELLO`.

BRIDGE_DEV_x_DEV_x_PORT_PRIORITY Default: `BRIDGE_DEV_x_DEV_x_PORT_PRIORITY='128'`

This setting is optional and can also be completely omitted.

Only valid if `BRIDGE_DEV_x_STP='yes'` is set!

Only relevant if multiple connections with the same `BRIDGE_DEV_x_DEV_x_PATHCOST` have the same destination. If this is the case the connection with lowest priority will be chosen.

Valid values are '0' to '240' in steps of '16'.

BRIDGE_DEV_x_DEV_x_PATHCOST Default: `BRIDGE_DEV_x_DEV_x_PATHCOST='100'`

This setting is optional and can also be completely omitted.

Only valid if `BRIDGE_DEV_x_STP='yes'` is set!

Indirectly specifies the bandwidth for this connection. The lower the value the higher is the bandwidth and therefore the connection gets a higher priority.

The calculation base proposed is 1000000 kbit/s which leads to the traffic costs listed in table 4.1. Please note to use the actual usable bandwidth in the formula when calculating. As a result this leads to significantly lower values than you would expect, especially on wireless lan.

Note: The current IEEE standard from 2004 uses 32 bit numbers for bandwidth calculation which is not supported on Linux yet.

Bandwidth	Setting of `BRIDGE_DEV_x_DEV_x_PATHCOST`
64 kbit/s	15625
128 kbit/s	7812
256 kbit/s	3906
10 Mbit/s	100
11 Mbit/s	190
54 Mbit/s	33
100 Mbit/s	10
1 Gbit/s	1

Table 4.1.: Values for BRIDGE_DEV_x_DEV_x_PATHCOST as a function of bandwidth

4.2.6. Notes

A bridge will forward any type of Ethernet data - thus e.g. a regular DSL modem can be used over WLAN as if it had a WLAN interface. No packets that pass the bridge will be examined for any undesirable activities (ie the fli4l packet filter is not active!). Use only after careful consideration of security risks (ie as a WLAN access point). There is also the possibility to activate EBTables support however.

4.2.7. EBTables - EBTables for fli4l

As of Version 2.1.9 fli4l has rudimental EBTables support. By setting `OPT_EBTABLES='yes'` EBTables support will get activated.

This means that all ebtables kernel modules get loaded and the ebtables program on the fli4l-routers will get available. In contrast to the much simplified netfilter configuration through the different filter lists of fli4l it then is necessary to write an ebtables script of your own. This means you have to write the complete ebtables script yourself.

For background informations about EBTables support please read the EBTables documentation at http://ebtables.sourceforge.net.

There is the possibility of issuing ebtables commands on the router before and after setting up the netfilter (`PF_INPUT_x`, `PF_FORWARD_x` etc). To do so, create the files ebtables.pre und

ebtables.post in the directory config/ebtables. Ebtables.pre will get executed before and ebtables.post after configuring the netfilter. Please remember that an error in the ebtables scripts may interrupt the boot process of the fli4l-router!

Before using EBTables you should definitely read the complete documentation. By using EBTables the complete behavior of the router may change! Especially filtering by mac: in PF_FORWARD will not work as before.

Have a look at this page giving a small glimpse about how the the ebtables support works: http://ebtables.sourceforge.net/br_fw_ia/br_fw_ia.html.

4.2.8. ETHTOOL - Settings for Ethernet Network Adapters

By setting `OPT_ETHTOOL='yes'` the ethtool program will be copied to the fli4l router in order to be used by other packages. By the help of this program, various settings of Ethernet network cards and drivers can be displayed and changed.

ETHTOOL_DEV_N Specify the number of settings to set at boot time.

Default: `ETHTOOL_DEV_N='0'`

ETHTOOL_DEV_x `ETHTOOL_DEV_x` indicates for which network device the settings should apply.

Example: `ETHTOOL_DEV_1='eth0'`

ETHTOOL_DEV_x_OPTION_N `ETHTOOL_DEV_x_OPTION_N` indicates the number of settings for the device.

ETHTOOL_DEV_x_OPTION_x_NAME

ETHTOOL_DEV_x_OPTION_x_VALUE The variable `ETHTOOL_DEV_x_OPTION_x_NAME` gives the name and `ETHTOOL_DEV_x_OPTION_x_VALUE` the value of the setting to be changed.

Following is a list of options and possible values activated by now:

- speed 10|100|1000|2500|10000 expandable by HD or FD (default FD = full duplex)
- autoneg on|off
- advertise %x
- wol p|u|m|b|a|g|s|d

Example:

```
OPT_ETHTOOL='yes'
ETHTOOL_DEV_N='2'
ETHTOOL_DEV_1='eth0'
ETHTOOL_DEV_1_OPTION_N='1'
ETHTOOL_DEV_1_OPTION_1_NAME='wol'
ETHTOOL_DEV_1_OPTION_1_VALUE='g'
ETHTOOL_DEV_2='eth1'
ETHTOOL_DEV_2_OPTION_N='2'
ETHTOOL_DEV_2_OPTION_1_NAME='wol'
ETHTOOL_DEV_2_OPTION_1_VALUE='g'
ETHTOOL_DEV_2_OPTION_2_NAME='speed'
ETHTOOL_DEV_2_OPTION_2_VALUE='100hd'
```

Further informations about ethtool can be found here: http://linux.die.net/man/8/ethtool

4.2.9. Example

For understanding a simple example is certainly helpful. In our example we assume 2 parts of a building which are connected by 2 x 100 Mbit/s lines. Four separate networks should be routed from one building to the other.

To achieve this a combination of bonding (joining the two physical lines) VLAN (to transport several separate networks on the bond) and bridging (to link the different nets to the bond/VLAN) is used. This has been tested successful on 2 Intel e100 cards and 1 Adaptec 4-port card ANA6944 in each building's router. The two e100 have the device names `'eth0'` and `'eth1'`. They are used for connecting the building. Intel e100's are the only cards known to work flawlessly with VLAN by now. Gigabit-cards should work too. The 4 ports of the multiport-card are used for the networks and have device names `'eth2'` to `'eth5'`.

At first the two 100 Mbit/s lines will be bonded:

```
OPT_BONDING_DEV='yes'
BONDING_DEV_N='1'
BONDING_DEV_1_DEVNAME='bond0'
BONDING_DEV_1_MODE='balance-rr'
BONDING_DEV_1_DEV_N='2'
BONDING_DEV_1_DEV_1='eth0'
BONDING_DEV_1_DEV_2='eth1'
```

This creates the device `'bond0'`. Now the two VLANs will be built on this bond. We use VLAN-IDs 11, 22, 33 und 44:

```
OPT_VLAN_DEV='yes'
VLAN_DEV_N='4'
VLAN_DEV_1_DEV='bond0'
VLAN_DEV_1_VID='11'
VLAN_DEV_2_DEV='bond0'
VLAN_DEV_2_VID='22'
VLAN_DEV_3_DEV='bond0'
VLAN_DEV_3_VID='33'
VLAN_DEV_4_DEV='bond0'
VLAN_DEV_4_VID='44'
```

Over this two VLAN connections the bridge into the networks segments will be built. Routing is not necessary this way.

```
OPT_BRIDGE_DEV='yes'
BRIDGE_DEV_N='4'
BRIDGE_DEV_1_NAME='_VLAN11_'
BRIDGE_DEV_1_DEVNAME='br11'
BRIDGE_DEV_1_DEV_N='2'
BRIDGE_DEV_1_DEV_1='bond0.11'
BRIDGE_DEV_1_DEV_2='eth2'
BRIDGE_DEV_2_NAME='_VLAN22_'
BRIDGE_DEV_2_DEVNAME='br22'
BRIDGE_DEV_2_DEV_N='2'
```

```
BRIDGE_DEV_2_DEV_1='bond0.22'
BRIDGE_DEV_2_DEV_2='eth3'
BRIDGE_DEV_3_NAME='_VLAN33_'
BRIDGE_DEV_3_DEVNAME='br33'
BRIDGE_DEV_3_DEV_N='2'
BRIDGE_DEV_3_DEV_1='bond0.33'
BRIDGE_DEV_3_DEV_2='eth4'
BRIDGE_DEV_4_NAME='_VLAN44_'
BRIDGE_DEV_4_DEVNAME='br44'
BRIDGE_DEV_4_DEV_N='2'
BRIDGE_DEV_4_DEV_1='bond0.44'
BRIDGE_DEV_4_DEV_2='eth5'
```

As a result all 4 Nets are connected with each other absolutely transparent and share the 200 Mbit/s connection. Even with a failure of one 100 Mbit/s line the connection will not fail. If necessary EBTables support can also be activated e.g. to activate certain packet filter.

This configuration is set up on two fli4l routers. I think this is an impressive example what the advanced_networking package can do.

4.3. CHRONY - Network Time Protocol Server/Client

OPT_CHRONY extends fli4l with the Network Time Protocol (Page 90) (NTP). Don't mix it up with the *normal* Time Protocol, the old OPT_TIME provides. The protocols aren't compatible wich possibly raises the need of new client-side programs, which understand NTP. If you can't abandon the simple time protocol, chrony can provide it as well. OPT_CHRONY works both as a server and client. Working as a client, OPT_CHRONY adjusts the time of the fli4l according to time referencies (time servers) in the internet. The basic setting uses up to three time servers from pool.ntp.org (Page 90). However it's possible to use a different selection of time servers in the configuration file. Thus it's possible to use time servers near you, if you choose de.pool.ntp.org if the router or your provider sits in Germany. For more details, look at the website of pool.ntp.org (Page 90).

Working as a client, OPT_CHRONY acts as a time reference for the local network (LAN). NTP uses port 123.

Chrony doesn't need a permanent connection to the internet. When the connection drops, chrony get's a notice and ceases the adjustion with the internet time servers. Neither does chrony dial to raise the connection nor prevent the automatic hangup, if `HUP_TIMEOUT`, the duration where no data is routet to the internet, is reached.

In order to do time adjustments smoothly, the following should be in mind:

- Chrony expects the BIOS-clock to be in UTC timezone. If not, it has to be configured in the configuration file.
  UTC = German time minus 1 (winter) or 2 (summer) hour(s)
- Since version 2.1.12 chrony sets the time during the first connection to the internet correct even if the time difference is huge (caused by an empty mainboard battery for example).
- If the BIOS isn't capable of handling years after 1999 (Year 2000 Bug) or if the BIOS clock is faulty, activate `OPT_Y2K='yes'` (Page 73)!

Only time servers in the internet which are reachable by the default route (`0.0.0.0/0`) can be used, because only the default route changes chrony into online mode. As an ethernet router with no DSL or ISDN circuits configured, chrony acts permanently in online mode.

Disclaimer: *The author gives neither a guarantee of functionality nor is he liable for any damage or the loss of data when using OPT_CHRONY.*

4.3.1. Configuration of OPT_CHRONY

The configuration is made, as for all fli4l packages, by adjusting the file `path/fli4l-3.10.5/<config>/chrony.txt` to meet the own demands. However almost all variables of OPT_CHRONY are optional. Optional means, the variables could, but need not be in the config file. Thus the chrony config file is almost empty and all optional variables have a usefull setting by default. To use different settings, the variables must be inserted into the config file by hand. Furthermore the description of every variable follows:

OPT_CHRONY Default: `OPT_CHRONY='no'`

The setting `'no'` deactivates OPT_CHRONY completely. There will be no changes made on the fli4l boot medium or the archive `opt.img`. Further OPT_CHRONY basically does not overwrite other parts of the fli4l installation with one exception. The paket filter file which is responsible for not counting the traffic coming from the outside (to make sure fli4l will drop the line for sure, when reaching the hangup time), will be exchanged. The new paket filter file ensures, that chrony-traffic doesn't count for the hangup time.
To activate OPT_CHRONY, the variable `OPT_CHRONY` has to be set to `'yes'`.

CHRONY_TIMESERVICE Default: `CHRONY_TIMESERVICE='no'`

With `CHRONY_TIMESERVICE` an additional protocol to transfer time informations can be invoked. It's only neccesary when the local hosts can't deal with NTP. It is conform to RFC 868 and uses port 37. If possible, prefer NTP.

Many thanks to Christoph Schulz, who provided the program `srv868`.

CHRONY_TIMESERVER_N Default: `CHRONY_TIMESERVER_N='3'`

`CHRONY_TIMESERVER_N` sets the number of reference time servers chrony uses. Add the same number of `CHRONY_TIMESERVER_x` variables. The index `x` must be increased up to the total number.
The basic setting uses up to three time servers from pool.ntp.org (Page 90).

CHRONY_TIMESERVER_x Default: `CHRONY_TIMESERVER_x='pool.ntp.org'`

With `CHRONY_TIMESERVER_x` an own list of internet time servers can be made. Either IP or DNS hostnames are possible.

CHRONY_LOG Default: `CHRONY_LOG='/var/run'`

`CHRONY_LOG` is the folder where chrony stores informations about the BIOS clock and the time adjustments. In almost any cases, this should not be altered.

CHRONY_BIOS_TIME Default: `CHRONY_BIOS_TIME='utc'`

To analyze the time of the BIOS clock (RTC = real time clock) correctly, `CHRONY_BIOS_TIME` provides if the clock runs at local `'local'` or universal time `'utc'` (UTC - Universal Coordinated Time).

4.3.2. Support

Support is only given in the fli4l Newsgroups (Page 90).

4.3.3. Literature

Homepage of chrony: http://chrony.tuxfamily.org/
NTP: The Network Time Protocol: http://www.ntp.org/
pool.ntp.org: public ntp time server for everyone: http://www.pool.ntp.org/en/
RFC 1305 - Network Time Protocol (Version 3) Specification, Implementation: http://www.faqs.org/rfcs/rfc1305.html
fli4l Newsgroups and the rules: http://www.fli4l.de/hilfe/newsgruppen/

4.4. DHCP_CLIENT - Dynamic Host Configuration Protocol

With this package, the router sources IP addresses for its interfaces dynamically. The package and its parameters are explained below.

4.4.1. OPT_DHCP_CLIENT

A DHCP client can be used to obtain an IP address for one or more interfaces of the router - this is most used with cable modem connections or in Switzerland, the Netherlands and France. Sometimes this configuration is also needed if the router is behind another router, that distributes the addresses via DHCP.

At the start of the router, for all specified interfaces IP addresses will be obtained. Subsequently, those are assigned to the interface and, if necessary, the default route on this interface is set.

OPT_DHCP_CLIENT Has to be set to 'yes', if one of the DHCP-clients is to be used.

Default Setting: `OPT_DHCP_CLIENT`='no'

DHCP_CLIENT_TYPE At the moment, the package is equipped with two different DHCP-clients, dhclient and dhcpcd. You can choose which one is to be used.

Default Setting: `DHCP_CLIENT_TYPE`='dhcpcd'

DHCP_CLIENT_N Here, the number of interfaces to be configured is neccesary.

DHCP_CLIENT_x_IF Here, the interface to be configured has to be specified as a reference to `IP_NET_x_DEV`, i.e. `DHCP_CLIENT_1_IF`='`IP_NET_1_DEV`'. The dhcp-client gets the apropriate device out of this variable. In base.txt a placeholder '*dhcp*' should be entered instead of an IP-address with netmask.

DHCP_CLIENT_x_ROUTE Here you can specify whether and how a route is to be set for the interface. The variable can take the following values:

none No route for the interface.

default Default route for the interface.

imond Imond is used to manage the default route for the interface.

Default Setting: DHCP_CLIENT_x_ROUTE='default'

DHCP_CLIENT_x_USEPEERDNS If this variable is set to 'yes' and the device has a default-route assigned, then the ISP's DNS server is used as a DNS forwarder on this route. Make sure to activate DNS forwarding - see base.txt.

Default Setting: DHCP_CLIENT_x_USEPEERDNS='no'

DHCP_CLIENT_x_HOSTNAME Some ISPs require a hostname to be forwarded. Ask your ISP for this and list it here. It does not have to be identical to the router hostname.

Default Setting: DHCP_CLIENT_x_HOSTNAME=''

DHCP_CLIENT_x_STARTDELAY This variable can optionally delay the start of the DHCP client. In some installations (eg fli4l as a dhcp client behind a cable modem, Fritzbox, a.s.o.) it is necessary to wait until the DHCP server in use is also started anew (if for example a power failure has happened).

Default Setting: DHCP_CLIENT_x_STARTDELAY='0'

DHCP_CLIENT_x_WAIT The DHCP client normally gets started as a background task. This means that the boot process is not delayed by the determination of the IPv4 address. Occasionally, however, it is necessary that the address is configured before the boot process progresses. This is the case when an installed package necessarily requires a configured address (eg in OPT_IGMP). Use DHCP_CLIENT_x_WAIT='yes' to force fli4l to wait for an address.

Default Setting: DHCP_CLIENT_x_WAIT='no'

DHCP_CLIENT_DEBUG Shows more informations during optainment of an address.

Default Setting: omit it or DHCP_CLIENT_DEBUG='no'

4.5. DNS_DHCP - Hostnames, DNS- and DHCP-Server as well as DHCP-Relay

4.5.1. Hostnames

Hosts

OPT_HOSTS The configuration of hostnames can be disabled by means of the optional variable OPT_HOSTS!

HOST_N HOST_x_{attribute} All hosts in the LAN should be described - with IP-address, name, aliasname and perhaps Mac-address for the dhcp-configuration. At first we have to set the number of computers with the variable HOST_N.

Note: Since version 3.4.0, the entry for the router comes from the information in the <config>/base.txt. For additional aliasnames, see HOSTNAME_ALIAS_N (Page 67).

Then the attributes define the properties of the hosts. Here are some of the attributes required, e.g. IP address and name, the other options are optional.

NAME – Name of the n'th host

IP4 – IP address (ipv4) of the n'th host

IP6 – IP address (ipv6) of the n'th host (optional). If you use "auto", then the address will be computed automatically – either the address `::ffff:<IPv4>` will be used, or (if you activate OPT_IPV6) a "proper" IPv6 address will be set, consisting of an IPv6 prefix (with /64 subnet mask) and the MAC address of the corresponding host. In order to make this work, you will have to set `HOST_x_MAC` (see below) and to properly configure the "ipv6" package.

DOMAIN – DNS domain of the n'th host (optional)

ALIAS_N – Number of aliasnames of the n'th host

ALIAS_m – m'th aliasname of the n'th host

MAC – MAC address of the n'th host

DHCPTYP – Assigning the IP address by DHCP depending on MAC or NAME (optional)

In the sample configuration file 4 hosts are configured, "client1", "client2", "client3" and "client4".

```
HOST_1_NAME='client1'                # 1st host: ip and name
HOST_1_IP4='192.168.6.1'
```

Alias names must be specified with complete domain.

The MAC address is optional and is only relevant if fli4l is used as a DHCP server additionally. This is explained below in the description of the optional package "`OPT_DHCP`" Without use as a DHCP Server only the IP address, the name of the host and possibly the alias name are used. The MAC address is a 48-bit address and consists of 6 hex values seperated by a colon, for example

```
HOST_2_MAC='de:ad:af:fe:07:19'
```

Note: If fli4l is supplemented with the IPv6 packet, IPv6 addresses are not needed, if the MAC addresses of the hosts are present, because the IPv6 packet calculates the IPv6 addresses automatically (modified EUI-64). Of course, you can disable the automatic and use dedicated IPv6 addresses, if you wish.

Extra Hosts

HOST_EXTRA_N HOST_EXTRA_x_NAME HOST_EXTRA_x_IP4 HOST_EXTRA_x_IP6
Using these variables, you may add other hosts which are not members of the local domain e.g. hosts on the other side of a VPN.

4.5.2. DNS-Server

OPT_DNS To activate the DNS-server the variable `OPT_DNS` must be set to 'yes'.

If in the LAN no Windows machines are used or it has already a running DNS server, `OPT_DNS` can be set to 'no' and you may skip the rest in this section.

In doubt, set (Default setting): `OPT_DNS`='yes'

General DNS-options

DNS_BIND_INTERFACES If you choose 'yes' here, dnsmasq does *not* listen on all IP-addresses and only binds and listens to IP-addresses defined in DNS_LISTEN. With option 'no' dnsmasq listens on all interfaces and IP-addresses, discarding request it normally should not react to. This is a problem if you like to use several DNS-servers on different IP-addresses. Additional DNS-servers on fli4l can be necessary i.e. if you need a slave-dns on fli4l. If you don't want to use dnsmasq exclusively on fli4l you have to choose 'yes' here and configure the IP-addresses for it via DNS_LISTEN.

DNS_LISTEN_N DNS_LISTEN_x If you chose `OPT_DNS='yes'`, use `DNS_LISTEN_N` to set the number and `DNS_LISTEN_1` up to `DNS_LISTEN_N` to specify the local IPs where **dnsmasq** accepts DNS-queries. If you set `DNS_LISTEN_N` to 0, **dnsmasq** answers DNS-queries on all local IPs. Only IPs of existing interfaces (ethernet, wlan ...) are allowed. Alternatively you can use ALIAS-Names here, i.e. `IP_NET_1_IPADDR`.

For all addresses specfied here ACCEPT rules will be created in the firewall's INPUT chains if `PF_INPUT_ACCEPT_DEF='yes'` and/or `PF6_INPUT_ACCEPT_DEF='yes'`. In case of `DNS_LISTEN='0'` rules allowing DNS access on *all* interfaces configured will be created.

If in doubt, the default settings should be used.

DNS_VERBOSE Logging of DNS-queries: 'yes' or 'no'

For detailed messages from the DNS `DNS_VERBOSE` has to be set to yes. DNS-queries are logged to the syslog then. To see the messages you must set `OPT_SYSLOGD='yes'` (Page 71) - see below.

DNS_MX_SERVER This variable indicates the hostname for the MX-record (Mail-Exchanger) for the domain defined in `DOMAIN_NAME`. A MTA (Mail"=Transport"=Agent, i.e. sendmail) on an internal server asks the DNS for a Mail-Exchanger for the destination domain of the mail beeing delivered.

This is no mail-client autoconfiguration for i.e. Outlook! So please do not insert gmx.de here and wonder why Outlook does not work.

DNS_FORBIDDEN_N DNS_FORBIDDEN_x Here you can provide domains, which are always beeing answered as "not existend".

Example:

```
DNS_FORBIDDEN_N='1'
DNS_FORBIDDEN_1='foo.bar'
```

In this case, a query for www.foo.bar is answered by an error. You can inhibit entire Top-Level-Domains in this way:

```
DNS_FORBIDDEN_1='de'
```

Then the name resolution for all hosts of the .de Topleveldomain is switched off.

DNS_REDIRECT_N DNS_REDIRECT_x DNS_REDIRECT_x_IP Here you can specify domains, which are beeing redirected to a specific IP.

Example:

```
DNS_REDIRECT_N='1'
DNS_REDIRECT_1='yourdom.dyndns.org'
DNS_REDIRECT_1_IP='192.168.6.200'
```

This redirects a query of yourdom.dyndns.org to IP 192.168.6.200.

DNS_BOGUS_PRIV If you set this variable to ‘yes‘, reverse-lookups for IP-Addresses of RFC1918 (Private Address Ranges) are not redirected to other DNS-servers but rather answered by the dnsmasq.

DNS_FILTERWIN2K If this is set to ’yes’ DNS queries of type SOA, SRV, and ANY will be blocked. Services using these queries will not work anymore without further configuration.
For example:

- XMPP (Jabber)
- SIP
- LDAP
- Kerberos
- Teamspeak3 (as of client-version 3.0.8)
- Minecraft (as of full version 1.3.1)
- domain controller discovery (Win2k)

For further information:

- Explanantion of DNS query types in general:
  http://en.wikipedia.org/wiki/List_of_DNS_record_types
- dnsmasq manpage:
  http://www.thekelleys.org.uk/dnsmasq/docs/dnsmasq-man.html
- SRV queries in detail:
  http://en.wikipedia.org/wiki/SRV_resource_record

By setting this to ’no’, additionally forwarded DNS queries may cause unwanted dial-up connections or prevent existing ones from being closed. Particularly if you are using ISDN or UMTS connections additional costs may arise. You have to choose for yourself what’s more important to you.

DNS_FORWARD_LOCAL By setting this variable to ’yes’ fli4l-routers may be configured to be in a domain by the name of DOMAIN_NAME=’example.local’ whose name resolution will be done by another name server specified by DNS_SPECIAL_x_DOMAIN=’example.local’.

DNS_LOCAL_HOST_CACHE_TTL defines the TTL (Time to live, in seconds) for entries defined in /etc/hosts as well as for hosts listed in DHCP. The default value for the fli4l-router is 60 seconds. Dnsmasq uses 0 as default and thus disables caching of DNS entries. The idea behind that is to reuse DHCP leases that are running out fastly and pass them on swiftly. However, if for example a local IMAP proxy queries the DNS entries several times per second this is a significant burden on the network. A compromise is a relatively short TTL of 60 seconds. Even without the short TTL 60 seconds a host can always simply be switched off, so that the polling software has to deal with hosts not responding anyway.

DNS_SUPPORT_IPV6 (optional)

Setting this optional variable to 'yes' enables the support for IPV6 Addresses of the DNS server.

DNS Zone Configuration

Dnsmasq can also manage a DNS domain autonomously, being "authoritative" for it. Two things have to be done to achieve this: At first you have to specify which external (!) DNS name service points to your fli4l and on which network interface the resolution takes place. The specification of an external reference is required because the domain that is managed by fli4l will always be a subdomain of another domain. [2] The specification of the "outward" interface is important because the dnsmasq will behave different from other "inward" interfaces there: "Outwards" dnsmasq will never answer queries for names outside of its configured own domain. "Inwards" dnsmasq also acts as a DNS relay to the Internet to accomplish resolution of non-local names.

As the second thing you have to configure which networks can be reached from outside via name resolution. Of course only nets with public IP addresses can be specified because hosts with private addresses cannot be reached from outside.

Below the configuration will be described with an example. This example assumes IPv6 packets as well as a publicly routed IPv6 prefix; the latter can be provided i.e. by a 6in4 tunnel provider such as SixXS or Hurricane Electric.

DNS_AUTHORITATIVE Specifying `DNS_AUTHORITATIVE='yes'` activates dnsmasq's authoritative mode. However, this is not enough, some more information must be given (see below).

Default Setting: `DNS_AUTHORITATIVE='no'`

Example: `DNS_AUTHORITATIVE='yes'`

DNS_AUTHORITATIVE_NS With this variable, the DNS name is configured by which the fli4l is referenced from outside using a DNS-NS record. This can also be a DNS name from a dynamic DNS service.

Example: `DNS_AUTHORITATIVE_NS='fli4l.noip.me'`

DNS_AUTHORITATIVE_IPADDR This variable configures the address resp. interface on which dnsmasq will answer DNS queries for your own domain authoritatively. Symbolic

[2] We assume that noone uses a fli4l as a DNS root server...

names like IP_NET_2_IPADDR are allowed. The dnsmasq can only answer authoritative on *one* address resp. interface.

Currently only permanently assigned addresses can be specified. Addresses derived only by a dial-in (for example, using a PPP connection), can not be used. This will be corrected in a later version of fli4l.

Important: *It should be noted that this should never be an address / interface connected to your own LAN, otherwise non-local names could not be resolved anymore!*

Example: DNS_AUTHORITATIVE_IPADDR='IP_NET_2_IPADDR'

DNS_ZONE_NETWORK_N DNS_ZONE_NETWORK_x Specify the network addresses here for which the dnsmasq should resolve names authoritatively. Both forward (name to address) and reverse lookup (address to name) will work.

A complete example:

```
DNS_AUTHORITATIVE='yes'
DNS_AUTHORITATIVE_NS='fli4l.noip.me'
DNS_AUTHORITATIVE_IPADDR='IP_NET_2_IPADDR' # Uplink connected to eth1
DNS_ZONE_NETWORK_N='1'
DNS_ZONE_NETWORK_1='2001:db8:11:22::/64'   # local IPv6-LAN
```

It is assumed here that "2001:db8:11::/48" is a IPv6 prefix publicly routed to fli4l and that subnet 22 was chosen for the LAN.

DNS Zone Delegation

DNS_ZONE_DELEGATION_N DNS_ZONE_DELEGATION_x There are special situations where the reference to one or more DNS server is useful, for example when using fli4l in an intranet without an Internet connection or a mix of these (intranet with an own DNS-server Internet connection in addition)

Imagine the following scenario:

- Circuit 1: Dial into the Internet
- Circuit 2: Dial into the Company network 192.168.1.0 (firma.de)

Then you set ISDN_CIRC_1_ROUTE to '0.0.0.0' and ISDN_CIRC_2_ROUTE to '192.168.1.0'. When accessing hosts with IP-Addresses with 192.168.1.x, fli4l will use circuit 2, otherwise circuit 1. But if the company network isn't public, it presumably has its own DNS server. Suppose the address of this DNS server would be 192.168.1.12 and the domain name would be " firma.de".

In this case you will write:

```
DNS_ZONE_DELEGATION_N='1'
DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_N='1'
DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_1_IP='192.168.1.12'
DNS_ZONE_DELEGATION_1_DOMAIN_N='1'
DNS_ZONE_DELEGATION_1_DOMAIN_1='firma.de'
```

Then, DNS queries for xx.firma.de are answered from the company's internal DNS server, otherwise the DNS server on the Internet is used

Another case:

- Circuit 1: Internet
- Circuit 2: Company network 192.168.1.0 *with* Internet-Access

Here we have two possibilities to reach the internet. To separate private from business, the following can be used:

```
ISDN_CIRC_1_ROUTE='0.0.0.0'
ISDN_CIRC_2_ROUTE='0.0.0.0'
```

We set a default route on both circuits and switch the route with the imond-client then - as desired. Also in this case set `DNS_ZONE_DELEGATION_N` and `DNS_ZONE_DELEGATION_x_DOMAIN_x` as described above.

If you want the reverse DNS resolution for such a network (e.g. an mail server will need this) you can provide the optional variable `DNS_ZONE_DELEGATION_x_NETWORK_x`, which lists the networks for active Reverse-Lookup. The following example illustrates this:

```
DNS_ZONE_DELEGATION_N='2'
    DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_N='1'
    DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_1_IP='192.168.1.12'
    DNS_ZONE_DELEGATION_1_DOMAIN_N='1'
    DNS_ZONE_DELEGATION_1_DOMAIN_1='firma.de'
    DNS_ZONE_DELEGATION_1_NETWORK_N='1'
    DNS_ZONE_DELEGATION_1_NETWORK_1='192.168.1.0/24'
    DNS_ZONE_DELEGATION_2_UPSTREAM_SERVER_N='1'
    DNS_ZONE_DELEGATION_2_UPSTREAM_SERVER_1_IP='192.168.2.12'
    DNS_ZONE_DELEGATION_2_DOMAIN_N='1'
    DNS_ZONE_DELEGATION_2_DOMAIN_1='bspfirma.de'
    DNS_ZONE_DELEGATION_2_NETWORK_N='2'
    DNS_ZONE_DELEGATION_2_NETWORK_1='192.168.2.0/24'
    DNS_ZONE_DELEGATION_2_NETWORK_2='192.168.3.0/24'
```

with the config option `DNS_ZONE_DELEGATION_x_UPTREAM_SERVER_x_QUERYSOURCEIP` you can define the source IP-address for outgoing DNS requests to upstream servers. This is useful i.e. if you reach the upstream DNS server via a VPN and and don't want the local VPN address of fli4l to appear as the source IP at the upstream server. Another usecase is an IP address not routable for the Upstream DNS server (could happen in a VPN). In this case it is as well necessary to set the IP address used by the dnsmasq to an IP used by fli4l to be accessible by the Upstream DNS Server.

```
DNS_ZONE_DELEGATION_N='1'
DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_N='1'
DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_1_IP='192.168.1.12'
DNS_ZONE_DELEGATION_1_UPSTREAM_SERVER_1_QUERYSOURCEIP='192.168.0.254'
DNS_ZONE_DELEGATION_1_DOMAIN_N='1'
DNS_ZONE_DELEGATION_1_DOMAIN_1='firma.de'
DNS_ZONE_DELEGATION_1_NETWORK_N='1'
DNS_ZONE_DELEGATION_1_NETWORK_1='192.168.1.0/24'
```

DNS_REBINDOK_N DNS_REBINDOK_x_DOMAIN The nameserver *dnsmasq* normally declines responses from other name servers containing IP addresses from private networks. It prevents a certain class of network attacks. But if you have a domain with private IP addresses and a separate name server that is responsible for this network, exactly the answers which would be rejected from *dnsmasq* are needed. List theese domains in `DNS_REBINDOK_x`, to accept answers from this domain.

Another example for nameservers delivering private IP-Addresses as an answer are so called "Real-Time Blacklist Server". An example based on these might look like this:

```
DNS_REBINDOK_N='8'
DNS_REBINDOK_1_DOMAIN='rfc-ignorant.org'
DNS_REBINDOK_2_DOMAIN='spamhaus.org'
DNS_REBINDOK_3_DOMAIN='ix.dnsbl.manitu.net'
DNS_REBINDOK_4_DOMAIN='multi.surbl.org'
DNS_REBINDOK_5_DOMAIN='list.dnswl.org'
DNS_REBINDOK_6_DOMAIN='bb.barracudacentral.org'
DNS_REBINDOK_7_DOMAIN='dnsbl.sorbs.net'
DNS_REBINDOK_8_DOMAIN='nospam.login-solutions.de'
```

4.5.3. DHCP-server

OPT_DHCP With `OPT_DHCP` you can activate the DHCP-server.

DHCP_TYPE (optional)

With this variable you can set if the internal DHCP-funktion of the dnsmasq should be used or if you want to use the external ISC-DHCPD. When using the ISC-DHCPD support for DDNS is not available.

DHCP_VERBOSE activates additional messages of DHCP in the log.

DHCP_LS_TIME_DYN determines the default lease-time for dynamically assigned IP-Addresses.

DHCP_MAX_LS_TIME_DYN determines the maximum lease-time for dynamically assigned IP-Addresses.

DHCP_LS_TIME_FIX Default lease-time for dynamically assigned IP-Addresses.

DHCP_MAX_LS_TIME_FIX determines the maximum lease-time for statically assigned IP-Addresses.

DHCP_LEASES_DIR Determines the folder for the leases-file. You may set an absolute path or *auto*. When *auto* is sued the lease file will be saved in a subdir of the persistent directory (see documentation for package base).

DHCP_LEASES_VOLATILE If the folder for the *Leases* resides inside the ram-disc (because the router boots i.e. from a CD or from another non-writeable device) the router will warn about a missing *Lease* file at each boot. This warning can be ommitted by setting `DHCP_LEASES_VOLATILE` to *yes*.

DHCP_WINSSERVER_1 sets the address of the first WINS-Server. If the WINS server is configured and activated in the SAMBA package the settings from there are used.

DHCP_WINSSERVER_2 sets the address of the second WINS-Server. If the WINS server is configured and activated in the SAMBA package the settings from there are used.

Local DHCP Ranges

DHCP_RANGE_N Number of DHCP ranges

DHCP_RANGE_x_NET Reference to one of the `IP_NET_x` networks

DHCP_RANGE_x_START sets the first IP-Address that can be used.

DHCP_RANGE_x_END sets the last assignable IP-Address. Both variables `DHCP_RANGE_x_START` and `DHCP_RANGE_x_END` could be left empty, so there will be no DHCP-Range, but hosts with MAC assignments will receive their values from the other variables.

DHCP_RANGE_x_DNS_SERVER1 sets the addresse of the DNS-server for DHCP-hosts of the network. This variable is optional. If left empty or omitted, the IP-address of the matching network is used. Further it's possible to set this variable to 'none'. Then, no DNS-server is assigned.

DHCP_RANGE_x_DNS_SERVER2 same settings for the second DNS-servers

DHCP_RANGE_x_DNS_DOMAIN sets a special DNS-domain for DHCP-hosts for this range. This variable is optional. If left empty or omitted, the default DNS-domain `DOMAIN_NAME` is used.

DHCP_RANGE_x_NTP_SERVER sets the address of the NTP-server for DHCP-hosts in this range. This variable is optional. If left empty or omitted, the IP-address of the `DHCP_RANGE_x_NET` network is used when a timeserver package is activated. If set to 'none', no NTP-Server is assigned.

DHCP_RANGE_x_GATEWAY sets the address of the gateway for this range. This variable is optional. If left empty or omitted, the IP-address of the `DHCP_RANGE_x_NET` network is used. If set to 'none', no gateway is assigned.

DHCP_RANGE_x_MTU sets the MTU for clients in this range. This variable is optional.

DHCP_RANGE_x_OPTION_N allows the setting of user defined options for this range. The available options are mentioned in the dnsmasq manual (http://thekelleys.org.uk/dnsmasq/docs/dnsmasq.conf.example). They are adopted unchecked - this could raise problems. This variable is optional.

Extra DHCP-Range

DHCP_EXTRA_RANGE_N sets the number of DHCP-ranges not assigned to local networks. For this a DHCP-relay has to be installed on the gateway of the remote network.

DHCP_EXTRA_RANGE_x_START first IP-address to be assigned.

DHCP_EXTRA_RANGE_x_END last IP-address to be assigned.

DHCP_EXTRA_RANGE_x_NETMASK Netmask for this range.

DHCP_EXTRA_RANGE_x_DNS_SERVER Address of the DNS-servers for this range.

DHCP_EXTRA_RANGE_x_NTP_SERVER Address of the NTP-server for this range.

DHCP_EXTRA_RANGE_x_GATEWAY Address of the default-gateway for this range.

DHCP_EXTRA_RANGE_x_MTU MTU for clients in this range. This variable is optional.

DHCP_EXTRA_RANGE_x_DEVICE Network interface over which this range can be reached.

Not allowed DHCP-clients

DHCP_DENY_MAC_N Number of MAC-Addresses of hosts which should be rejeced.

DHCP_DENY_MAC_x MAC-Address of the host which should be rejeced.

Support For Network Booting

The dnsmasq supports clients booting by Bootp/PXE over the network. The needed informations for this are provided by dnsmasq and are configured per subnet and host. The needed variables are in the DHCP_RANGE_%- and HOST_%-sections and point to the bootfile (*_PXE_FILENAME), the server which hosts this file (*_PXE_SERVERNAME and *_PXE_SERVERIP) and perhaps necessary options (*_PXE_OPTIONS). Furthermore the internal tftp-server can be activated to provide network booting entirely from dnsmasq.

HOST_x_PXE_FILENAME DHCP_RANGE_x_PXE_FILENAME The bootfile. If PXE is used, the pxe-bootloader, i.e. pxegrub, pxelinux or similar.

HOST_x_PXE_SERVERNAME HOST_x_PXE_SERVERIP DHCP_RANGE_x_PXE_SERVERNAME Name and IP of the tftp-servers. If empty, the router itself is used.

DHCP_RANGE_x_PXE_OPTIONS HOST_x_PXE_OPTIONS Some bootloader need special options to boot. I.e. pxegrub asks by use of option 150 for the name of the menu file. This options can be put here. For pxegrub it looks like this:

```
HOST_x_PXE_OPTIONS='150,"(nd)/grub-menu.lst"'
```

If more options are needed, separate them by a space.

4.5.4. DHCP-Relay

A DHCP-relay is used, when another DHCP-Server manages the ranges which is not directly reachable from the clients.

OPT_DHCPRELAY Set to 'yes' to act as a DHCP-relay. To act as a DHCP-server is not allowed at the same time.

Default setting: `OPT_DHCPRELAY='no'`

DHCPRELAY_SERVER Insert the right DHCP-server to which the queries should be forwarded.

DHCPRELAY_IF_N DHCPRELAY_IF_x Sets `DHCPRELAY_IF_N` the number of network interfaces, the relay-server listens to. In `DHCPRELAY_IF_x` provide the apropriate network interfaces.

Provide the interface the DHCP-server uses to answer as well. Make sure to set the routes on the CP running the DHCP server are correct. The answer of the DHCP server is deirected to the interface to which the clioent is connected.

Assume the following szenario:

- relay with two interfaces
- interface to the clients: eth0, 192.168.6.1
- interface to the DHCP-server: eth1, 192.168.7.1
- DHCP-server: 192.168.7.2

A route on the DHCP server has to exist over which the answers to 192.168.6.1 can reach their destination. If the router on which the relay is running is the default gateway for the DHCP server everything os fine already. If not, an extra route is needed. If the DHCP server is a fli4l ythe following config variable is sufficient: IP_ROUTE_x='192.168.6.0/24 192.168.7.1'

There may be warnings about ignoring certain packets which you may safely ignore.

Example:

```
OPT_DHCPRELAY='yes'
DHCPRELAY_SERVER='192.168.7.2'
DHCPRELAY_IF_N='2'
DHCPRELAY_IF_1='eth0'
DHCPRELAY_IF_2='eth1'
```

4.5.5. TFTP-server

To deliver files with the TFTP-protocol, a TFTP-server is needed. This may be useful for netboot scenarios.

OPT_TFTP Activates the internal TFTP-server of the dnsmasq. Default setting is 'no'.

TFTP_PATH Specifies the folder, where the files to be delivered to the clients are stored. The files have to be stored manually there.

4.5.6. YADIFA - Slave DNS Server

OPT_YADIFA Activates the YADIFA Slave DNS Server. Default Setting: 'no'.

OPT_YADIFA_USE_DNSMASQ_ZONE_DELEGATION If this setting is activated the yadifa start script will automatically generate the according zone delegation entries for dnsmasq. The slave zones can be queried directly from dnsmasq and basically YADIFA_LISTEN_x entries not needed. Queries will only be forwarded to yadifa which is listening on localhost:35353 and then answered by dnsmasq.

YADIFA_LISTEN_N If you specified `OPT_YADIFA`='yes' you may provide local IPs on which yadifa is allowed to answer queries. `YADIFA_LISTEN_N` sets the number and `YADIFA_LISTEN_1` to `YADIFA_LISTEN_N` the local IPs. A port number is otional, with 192.168.1.1:5353 teh YADIFA Slave DNS Server would listen to DNS queries on port 5353. Please note that dnsmasq is not allowed to listen on all interfaces in this case (see `DNS_BIND_INTERFACES`). Only IPs of existing interfaces may be used here (ethernet, wlan ...) otherwise there will be warning during router boot. As an alternative it is possible to use an ALIAS name, like i.e. `IP_NET_1_IPADDR`

YADIFA_ALLOW_QUERY_N

YADIFA_ALLOW_QUERY_x Sets the IP addresses and nets that are allowed to access YADIFA. This setting will be used by YADIFA to configure fli4l's packet filter accordingly and to generate the configuration files for YADIFA. By the prefix "!" acces to YADIFA is denied for the IP address or network in question.

The fli4l packet filter will be configured in a way that all nets allowed in this variable and those for the zones are joined in an ipset list (yadifa-allow-query). A differentiation on zones is not possible for the packet filter. In addition all IP addressesand nets from this global setting whose access is denied will be added to the list. So you can't reenable access later on.

YADIFA_SLAVE_ZONE_N Specifies the number of slave DNS zones YADIFA should take care of.

YADIFA_SLAVE_ZONE_x The name of the slave DNS zone.

OPT_YADIFA_SLAVE_ZONE_USE_DNSMASQ_ZONE_DELEGATION Activates ('yes') or deactivates ('no') the dnsmasq zone delegation only for the slave zone.

YADIFA_SLAVE_ZONE_x_MASTER The IP address of the DNS master server with an optional port number.

YADIFA_SLAVE_ZONE_x_ALLOW_QUERY_N

YADIFA_SLAVE_ZONE_x_ALLOW_QUERY_x Specifies IP addresses and nets for which access to this YADIFA DNS zone is allowed. This can be used to limit access to certain DNS zones even more. YADIFA uses this setting to generate its configuration files.

By the prefix "!" acces to YADIFA is denied for the IP address or network in question.

4.6. DSL - DSL over PPPoE, Fritz!DSL and PPTP

fli4l supports DSL in three different variants:

- PPPoE (external DSL-modems connected over ethernet using pppoe)
- PPTP (external DSL-modems connected over ethernet using pptp)
- Fritz!DSL (DSL over DSL-adapters manufactured by AVM)

You can choose only one of these options, simultaneous operation isn't possible yet. The configuration for all variants is similar, so the general parameters are described at first and then the special options for the individual variants will be discussed. DSL-access is handled by imond as a circuit. Therefore it is necessary to activate imond by setting (see START_IMOND (Page 67)) to `'yes'`.

4.6.1. General Configuration Variables

The packages all use the same configuration variables, they differ only by the package name prefixes. As an example: in all packages the user name is required. The variable is named `PPPOE_USER`, `PPTP_USER` or `FRITZDSL_USER` depending on the package. The variables are described indicating the missing prefix by an asterisk. In concrete examples PPPOE is assumed but these are valid with any other prefix too.

***_NAME** Set a name for the circuit - max. 15 digits long. This will be shown in the imon client imonc. Spaces (blanks) are not allowed.

Example: `PPPOE_NAME='DSL'`

***_USEPEERDNS** This specifies whether the address of the name servers given by the internet service provider at dial-in will be added to the configuration file of the local name server for the duration of the online connection or not.

It only makes sense to use this option for internet access circuits. Almost all providers support these type of transfer by now.

After the name server IP addresses have been transferred, name servers in `DNS_FORWARDERS` will be used as forwarders. Afterwards the local nameserver is forced to reread its configuration. Previously resolved names will not be lost from the name server cache.

This option has the advantage of always using the nearest name servers (given the provider transmits the correct IP addresses) thus accelerating the name resolution process.

In case of failure of a provider's DNS server in most cases the incorrect DNS server addresses are rapidly corrected by the provider.

Nevertheless it is necessary to specify a valid name server in `DNS_FORWARDERS`. Otherwise the first name request can not be resolved correctly. In addition the original configuration of the local nameservers will be restored when terminating the connection.

Default-setting: `*_USEPEERDNS='yes'`

***_DEBUG** If you want pppd to be more verbose set `*_DEBUG` to `'yes'`. In this case pppd will write additional informations to syslog.

Attention: In order to get this to work syslogd has to activated by setting `OPT_SYSLOGD` to `'yes'` in base.txt.

***_USER, *_PASS** Provide user ID an passwort for the provider used. `*_USER` is used for the user id `*_PASS` is the password.

Attention: For T-Online-access consider this:

Username AAAAAAAAAAAATTTTTT#MMMM is put together from a 12 digit 'Anschlußkennung', T-Online-number and 'Mitbenutzernummer'. Behind the T-Online-number follows a '#' (only if its length is shorter than 12 digits).

If this does not work in rare cases put another '#' between 'Anschlußkennung' and T-Online-number.

User ID for T-Online has to have an additional '@t-online.de' at the end!

Example:

```
PPPOE_USER='11111111111222222#0001@t-online.de'
```

Infos on user ID's for other providers are found in the FAQ:

- http://extern.fli4l.de/fli4l_faqengine/faq.php?list=category&catnr=3&prog=1

***_HUP_TIMEOUT** Specify the time in seconds after which the connection will be terminated when no more traffic is detected over the DSL line. A timeout of '0' or 'never' stands for no timeout. Using 'never' the router immedeately reconnects after a disconnection. Changing dialmodes is not possible then - it has to stay 'auto'. 'Never' is currently only available for PPPOE and FritzDSL.

***_CHARGEINT** Charge-interval: Time interval in seconds which will be used for calculating online costs.

Most provider calculate their charges per minute. In this case put in '60'. For providers that charge per second set `*_CHARGEINT='1'`.

Unfortunately for DSL the timing is not fully utilized, as it is the case for ISDN. In our case a hangup will be triggered after the time specified in `*_HUP_TIMEOUT`.

Hence `*_Chargeint` is only significant for the calculation of charges.

***_TIMES** This times determine when to enable the circuit and how much it will cost. This makes it possible to have different default routes at different times for a circuit (Least Cost Routing). The imond daemon controls route assignment then.

Composition of variables:

```
PPPOE_TIMES='times-1-info [times-2-info] ...'
```

Each field times-?-info consists of 4 parts divided by colons (':').

1. Field: W1-W2

   Weekday-period, i.e. Mo-Fr or Sa-Su aso. English and german notations are both valid. For a single weekday write W1-W1 (i.e. Su-Su).

2. Field: hh-hh

   Hours, i.e. 09-18 or 18-09 too. 18-09 is equivalent to 18-24 plus 00-09. 00-24 means the complete day.

3. Field: Charge

   Currency-values as costs per minute, i.e. 0.032 for 3.2 Cent per minute. They are used to calculate the actual costs incurred, taking into account the cycle time. They will be displayed in the imon client.

4. Field: LC-Default-Route

   Content can be Y or N, which means:

   Y: The specified time range will be used as default route for LC-routing.

   N: The specified time range is only used for calculating costs but it won't be used for automatic LC-routing.

Example (read as one long line):

```
PPPOE_TIMES='Mo-Fr:09-18:0.049:N
             Mo-Fr:18-09:0.044:Y
             Sa-Su:00-24:0.039:Y'
```

Important: *Times used in `*_TIMES` have to cover the whole week. If that is not the case a valid configuration can't be build.*

Important: *If the time ranges off all LC-default-route-circuits ("Y") togethter don't contain the complete week there will be no default route in these gaps. There will be no internet access possible at these times!*

One more simple example:

```
PPPOE_TIMES='Mo-Su:00-24:0.0:Y'
```

for those using a flatrate.

One last comment to LC-routing: *holidays are treated as Sundays.*

***_FILTER** fli4l hangs up automatically if there is no traffic over the pppoe interface in the time specified in hangup timeout. Unfortunately also traffic from the outside counts here, for example connection attempts by a P2P client such as eDonkey. Since you will be contacted almost permanently from outside nowadays, it can happen that fli4l never terminates the DSL connection.

Option `*_FILTER` comes to help here. Setting it to yes will only consider traffic that is generated from your own machine and external traffic will be completely ignored. Since incoming traffic usually means that the router or computers behind it respond by i.e rejecting requests additionally some outgoing packets are ignored. Read here how this exactly works:

- http://www.fli4l.de/hilfe/howtos/basteleien/hangup-problem-loesen/ and
- http://web.archive.org/web/20061107225118/http://www.linux-bayreuth.de/dcforum/DCForumID2/46.html.

A more detailed description of the expression and its usage is to be found in the appendix but is only interesting if you want to make changes.

***_FILTER_EXPR** Filter to use if `*_FILTER` is set to `'yes'`.

***_MTU *_MRU** These variables are optional and can be completely omitted.

With these optional variables the so-called **MTU** (maximum transmission unit) and the **MRU** (maximum receive unit) can be set. By default MTU and MRU are set to 1492. This setting should only be changed in special cases! These variables don't exist for OPT_PPTP.

***_NF_MSS** For some providern these effects can occur:

- Webbrowser gets a connection but is unresponsive after that,
- small mails can be sent but big mails can't,
- ssh works, scp hangs after initial connecting.

To work around this problems fli4l manipulates the MTU as a default. In some cases this is not enough so fli4l explicitely permits setting of the MSS (message segment size) to a value given by the provider. If the provider does not give any values 1412 is a good start to try. If a MTU is given by the provider, subtract 40 Byte here ($mss = mtu - 40$). These variables are optional and can be completely omitted. These variable doesn't exist for OPT_PPTP.

4.6.2. OPT_PPPOE - DSL over PPPoE

For communication via a DSL connection usually the PPPoE package is necessary because the provider does not provide a full blown router, but only a DSL modem. Between the fli4l router and the modem the PPP protocol is used, but in this case specifically over Ethernet.

One or two ethernet cards can be used in fli4l:

- only one card with IP for the LAN and PPP towards the DSL-modem
- two cards: one with IP for the LAN, the other for PPP towards the DSL-modem

The better choice is to use two ethernet cards. This way both protocols - IP and PPPoE - are clearly separated from each other.

But the method with one ethernet cards works as well. In this case the DSL-modem has to be connected to the network hub like all other clients. The maximum transfer speed can be slightly affected this way.

If experiencing communication problems between the modem and the network card you can try to slow down the speed of the NIC, eventually even switching it to half-duplex mode. All PCI cards but only a few PCI-ISA network cards can be configured to run in various speed modes. Either use ethtool from the package advanced_networking or create a DOS boot media and include the native configuration tool there. Start fli4l with this media and execute the card's native tool to choose and save the slower operation mode to the card. The configuration tool usually is available on the driver media or can be downloaded from the website of the card manufacturer. You may also find it in a search in the wiki:

- https://ssl.nettworks.org/wiki/display/f/Netzwerkkarten

If using two network cards you should use the first for the LAN and the second for the connection to the DSL modem.

Only the first card does need an IP address.

This means:

```
IP_NET_N='1'                # Only *one* card with IP-address!
IP_NET_1xxx='...'             # Usual parameters
```

For `PPPOE_ETH` set the second ethernet card to 'eth1' and define *no* `IP_NET_2-xxx`-variables.

OPT_PPPOE activates support for PPPoE. Default setting: `OPT_PPPOE='no'`.

PPPOE_ETH Name of the ethernet interface

'eth0'	first ethernet card
'eth1'	second ethernet card
...	...

Default setting: `PPPOE_ETH='eth1'`

PPPOE_TYPE *PPPOE* stands for transmission of PPP-packets over ethernet lines. Data to be transmitted is transformed to ppp-packets in a first step and then in a second step wrapped in pppoe-packets to be transmitted over ethernet to the DSL-modem. The second step can be done by the pppoe-daemon or by the kernel. `PPPOE_TYPE` defines the way of pppoe packet generation.

Values	Description
async	Packets are genereated by the pppoe-daemon; asynchronous communication between *pppd* and *pppoed*.
sync	Packets are genereated by the pppoe-daemon; synchronous communication between *pppd* and *pppoed*. This way communication is more efficient and thus leads to lower processor load.
in_kernel	Packets are genereated by the linux kernel. This way communication with a second daemon is omitted saving a lot of in-memory copying thus leading to even lower processor load.

Table 4.2.: Ways of generating pppoe packets

Somebody did a comparison between the various variants on a Fujitsu Siemens PCD-H, P75 the results are shown in table 4.3 [3].

PPPOE_HUP_TIMEOUT Using PPPoE-type in_kernel and dialmode auto, timeout can be set to 'never'. The router then no longer hangs up and after a forced disconnection by the provider dials again automatically. Subsequent changing of the dialmode is not possible anymore.

[3] Values were taken from a posting on spline.fli4l and have not been evaluated further. Message ID of the article: <caf9fkala1@bla.spline.inf.fu-berlin.de>.

fli4l	NIC	Bandwidth (downstream)	CPU-load
2.0.8	rtl8029 + rtl8139	310 kB/s	100%
2.0.8	2x 3Com Etherlink III	305 kB/s	100%
2.0.8	SMC + 3Com Etherlink III	300 kB/s	100%
2.1.7	SMC + 3Com Etherlink III	375 kB/s	40%

Table 4.3.: Bandwidth und CPU-load with pppoe

4.6.3. OPT_FRITZDSL - DSL via Fritz!Card DSL

Internet connection by Fritz! Card DSL is activated here. A Fritz! Card DSL by AVM is used for the internet connection. Since the drivers for these cards are not subject to the GPL it is not possible to provide them with the DSL package. It is essential to download these drivers before from http://www.fli4l.de/download/stabile-version/avm-treiber/ and to extract them into the fli4l directory.

Circuit-support for Fritz!Card DSL was realised with friendly help from Stefan Uterhardt (email: zer0@onlinehome.de).

OPT_FRITZDSL activates support for Fritz!DSL. Default setting: `OPT_FRITZDSL='no'`.

FRITZDSL_TYPE Several Fritz!-cards exist for a DSL connection. The card used will be specified by `FRITZDSL_TYPE`, have a look at table 4.4 for enumerating the supported cards.

Card Type	Usage
fcdsl	Fritz!Card DSL
fcdsl2	Fritz!Card DSLv2
fcdslsl	Fritz!Card DSL SL
fcdslusb	Fritz!Card DSL USB
fcdslslusb	Fritz!Card DSL SL USB
fcdslusb2	Fritz!Card DSL USBv2

Table 4.4.: Fritz-Cards

Default setting:

```
FRITZDSL_TYPE='fcdsl'
```

FRITZDSL_PROVIDER With this option the type of the remote station is set. Possible options are:
U-R2, ECI, Siemens, Netcologne, oldArcor, Switzerland, Belgium, Austria1, Austria2, Austria3, Austria4

In Germany almost always UR-2 i used. Siemens and ECI are only used with very old ports.
For Switzerland and Belgium, the options are self-explanatory and in Austria you have to try what works for you.
If anyone has a better description for the options in Austria please tell.

Default setting:

```
FRITZDSL_PROVIDER='U-R2'
```

4.6.4. OPT_PPTP - DSL over PPTP in Austria/the Netherlands

In Austria (and other european countries) PPTP-protocol is used instead of PPPoE. A separate ethernet card is connected to a PPTP-Modem in this case.

As of version 2.0 connecting via PPTP is realised as a Circuit - with the friendly help of Rudolf Hämmerle (email: rudolf.haemmerle@aon.at).

PPTP uses two cards. The first card should be used for connecting the LAN and the second for connecting to the DSL-modem.

Only the first card can have an IP-address.

This means:

```
IP_NET_N='1'                    # Only *one* card with IP-address!
IP_NET_1xxx='...'               # the usual parameters
```

`PPTP_ETH` is set to 'eth1' for the second ethernet card and *no* `IP_NET_2-xxx`-variables are defined.

OPT_PPTP activates support for PPTP. Default setting: `OPT_PPTP='no'`.

PPTP_ETH Name of the ethernet interfaces

'eth0'	first ethernet card
'eth1'	second ethernet card
...	...

Default setting: `PPTP_ETH='eth1'`

PPTP_MODEM_TYPE There are several PPTP modem types to realise a pptp connection. Define `PPTP_MODEM_TYPE` to define the modem type. Possible types are shown in table 4.5.

Modem Type	Usage
telekom	Austria (Telekom Austria)
xdsl	Austria (Inode xDSL@home)
mxstream	the Netherlands, Danmark

Table 4.5.: PPTP Modem Types

Default setting:

```
PPTP_MODEM_TYPE='telekom'
```

Inode xDSL@home

Support for Inode xDSL@home was implemented using information found on Inode's support pages[4].

[4] See http://www6.inode.at/support/internetzugang/xdsl_home/konfiguration_ethernet_linux.html

At the moment there are sometimes problems with renewing of leases for the interface (the IP for the interface is delivered by dhcp and has to be renewed in regular periods). Hanging up and reconnecting via imonc does not work well by now. Help by providing patches or as a tester is highly appreciated.

With xdsl two further options exist for pptp:

PPTP_CLIENT_REORDER_TO The pptp-client which is used for xdsl under certain circumstances must rearrange and buffer packets. It usually waits 0.3 seconds for a packet. By setting this variable you can vary the timeout between 0.00 (no buffer) and 10.00. Times always must be provided with two decimales.

PPTP_CLIENT_LOGLEVEL Here you can define how much debug output will be produced by the pptp-client. Values are 0 (little), 1 (default) und 2 (much).

4.6.5. OPT_POESTATUS - PPPoE-Status-Monitor On fli4l-Console

PPPoE-Status-Monitor for DSL Connections was developed by Thorsten Pohlmann.

With setting `OPT_POESTATUS='yes'` dsl status can be watched on the third fli4l console at any time. Switch to the third console by pressing ALT-F3 and back to the first console with ALT-F1.

4.7. DYNDNS - Dynamic Update For Domain Name Services

This package is used to update a dynamic hostname at every dial-in process. The following services are supported:

Provider	FreeDNS (afraid.org)
DYNDNS_x_PROVIDER	AFRAID
Homepage	http://freedns.afraid.org

Important: *Provide the last part of the URL downloadable on Afraid.org's homepage (behind the question mark) as password (Login ⇒„Dynamic DNS" ⇒The URL behind the Link „Direct URL"). All other informations will be ignored.*

Provider	Companity
DYNDNS_x_PROVIDER	COMPANITY
Homepage	http://www.staticip.de/

Provider	DHS International
DYNDNS_x_PROVIDER	DHS
Homepage	http://www.dhs.org/

Provider	DNS2Go
DYNDNS_x_PROVIDER	DNS2GO
Homepage	http://dns2go.com/

Provider	DNS-O-Matic
DYNDNS_x_PROVIDER	DNSOMATIC
Homepage	http://www.dnsomatic.com

Provider	DtDNS
DYNDNS_x_PROVIDER	DTDNS
Homepage	http://www.dtdns.com/

Provider	DynAccess
DYNDNS_x_PROVIDER	DYNACCESS
Homepage	http://dynaccess.de/

Important: *DynAccess offers special charges for the subdomains *.dyn-fli4l.de, *.dyn-fli4l.info and *.dyn-eisfair.de because of a fli4l-DynAccess cooperation. Informations can be found here:* http://www.dyn-fli4l.de/ *or* http://www.dyn-eisfair.de/.

Provider	DynDNS.org
DYNDNS_x_PROVIDER	DYNDNS
Homepage	http://dyn.com/

Provider	DynDNS.org (custom)
DYNDNS_x_PROVIDER	DYNDNSC
Homepage	http://dyn.com/standard-dns/

Provider	DynDNS DK
DYNDNS_x_PROVIDER	DYNDNSDK
Homepage	http://dyndns.dk/

Provider	dyndns:free
DYNDNS_x_PROVIDER	DYNDNSFREE
Homepage	http://dyndnsfree.de/

Provider	eisfair.net
DYNDNS_x_PROVIDER	DYNEISFAIR
Homepage	http://www.intersales.de/it-infrastruktur/dyneisfair.html

Important: *By using this service you support the work of the fli4l- and eisfair-developers.*

Provider	DyNS
DYNDNS_x_PROVIDER	DYNSCX
Homepage	http://www.dyns.cx/

Provider	GnuDIP Dynamic DNS
DYNDNS_x_PROVIDER	GNUDIP
Homepage	http://gnudip2.sourceforge.net/

Provider	Provider Hurricane Electric
DYNDNS_x_PROVIDER	HE
Homepage	https://dns.he.net/

Provider	IN-Berlin e.V.
DYNDNS_x_PROVIDER	INBERLIN
Homepage	http://www.in-berlin.de

Provider	KONTENT
DYNDNS_x_PROVIDER	KONTENT
Homepage	http://www.kontent.de/

Provider	Nerdcamp.net
DYNDNS_x_PROVIDER	NERDCAMP
Homepage	http://nerdcamp.net/dynamic/dns.cgi

Provider	No-IP.com
DYNDNS_x_PROVIDER	NOIP
Homepage	http://www.no-ip.com/

Provider	noxaDynDNS
DYNDNS_x_PROVIDER	NOXA
Homepage	http://www.noxa.de/

Provider	OVH.DE
DYNDNS_x_PROVIDER	OVHDE
Homepage	http://www.ovh.de/

Provider	PHPDYN
DYNDNS_x_PROVIDER	PHPDYN
Homepage	http://www.webnmail.de/phpdyn/

Important: *you have to host this type for yourself*

Provider	Regfish.com
DYNDNS_x_PROVIDER	REGFISH
Homepage	http://www.regfish.de/

Provider	SelfHost.de
DYNDNS_x_PROVIDER	SELFHOST
Homepage	http://selfhost.de/cgi-bin/selfhost

Provider	Securepoint Dynamic DNS Service
DYNDNS_x_PROVIDER	SPDNS
Homepage	http://www.spdns.de/

Provider	Strato
DYNDNS_x_PROVIDER	STRATO
Homepage	http://www.strato.de/

Provider	T-Link.de
DYNDNS_x_PROVIDER	TLINK
Homepage	http://www.t-link.de/

Provider	twodns.de
DYNDNS_x_PROVIDER	TWODNS
Homepage	http://www.twodns.de/

Provider	ZoneEdit.com
DYNDNS_x_PROVIDER	ZONEEDIT
Homepage	http://zoneedit.com/

We try to keep this list up-to-date but do not rely too closely on it. There is no liability at all for the correctness of this data. You may report errors or changes detected by sending a mail to the fli4l team (email: team@fli4l.de).

This is a complete list. Other providers are not supported without changes. How to expand the package to support new providers can be found in the appendix.

The dynamic hostname will be actualized with every internet dial-in. The package includes a lock that prevents repeated updates of the same IP as this is frowned with some DynDNS providers and in extreme cases can lead to account blocking.

Note: The changing of the dynamic hostname may take some minutes.

Before using the package you have to aquire an account with one of the providers named above. If you already have an account you are ready to start. If you have no account yet, you can be guided by the table above to find a host name which fulfills the requirements and meets the personal taste.

For the configuration you will need the following data:

- Name of the provider
- Username
- Password
- DynDNS-hostname

The Data my vary with the provider while we try to provide a consistent configuration. Sometimes the hostname equals to the username, in such cases we will try to use the Host-field and ignore the username.

OPT_DYNDNS Setting this to `'yes'` activates `OPT_DYNDNS`.

DYNDNS_SAVE_OUTPUT By activating this with `'yes'` the result of the DynDNS query will be saved to a file shown by the webserver [5]

DYNDNS_N Change this value if you have accounts with more DynDNS providers and therefore want to update several names with every dial-in.

DYNDNS_x_PROVIDER Specify the name of the provider (see table above and hints in the config file).

DYNDNS_x_USER Your username for the DynDNS provider. Mostly an email address, a name you choose for yourself when registering or the DynDNS hostname.

DYNDNS_x_PASSWORD Your passwort for the DynDNS account. Be aware to keep your secret!

DYNDNS_x_HOSTNAME The *complete* DynDNS hostname for the account. Examples:

- `cool.nerdcamp.net`
- `user.dyndns.org`
- `fli4luser.fli4l.net`

[5] OPT_HTTPD

DYNDNS_x_UPDATEHOST For the provider PHPDYN specify here on which host the updater is installed. You need this because of PHPDYN not being a real provider but only a gpl'd script actualizing a PowerDNS Server with MySQL.

DYNDNS_x_CIRCUIT Set the circuits for which dialing in should update the hostname. Circuits are separated by spaces. It may be useful to update hostnames only when dialing in via DSL. Some examples:

```
DYNDNS_1_CIRCUIT='1 2 3'          # Only ISDN: Circuits 1 to 3
or
DYNDNS_1_CIRCUIT='pppoe'          # Only DSL: pppoe-Circuit
or
DYNDNS_1_CIRCUIT='dhcp'           # Update with DHCP providers
                                  # (opt_dhcp needed)
or
DYNDNS_1_CIRCUIT='pppoe 1'        # DSL and ISDN

or
DYNDNS_1_CIRCUIT='static'         # fli4l i.e. behind a LTE Router
```

DYNDNS_x_RENEW Some providers expect an update every n days even without your IP having changed. This interval may be set here. If no value is given an update will be forced every 29 days.

It should be noted that an update is triggered only when dialing. This means: DSL or ISDN connections or a renewing of a lease of an interface configured via DHCP like in a cable modem. If no dialing occurs for a longer time period you have to trigger the update in other ways.

DYNDNS_x_EXT_IP This variable configures the method by which the external IP address is detected. By setting this to `'no'` no external service will be queried for the IP address and the IP address of the WAN interface will be used instead. This usually works only with WAN connections which terminate directly on the fli4l, such as PPPoE via DSL. If specifying `'dyndns'` the IP address used when updating is determined by using the external service of checkip.dyndns.org. When using `'stun'` the list of STUN servers is queried one by one until a valid response is returned. The use of an external service to determine the IP address is necessary if the router itself is not getting the external IP. It should be noted that the router will not note a change of the external IP in this case and thus can't update the dyndns name entry immedeately.

DYNDNS_x_LOGIN Some providers require the user to log in to their Web page regularly by using her/his user account to prevent the service from being deactivated. If this variable is set to `'yes'`, the FLI4L will do that for you. However, this only works if the dyndns package is prepared for this provider. Such a regular registration is currently only necessary and possible for the provider "DYNDNS". Please remember also that the use of this function requires `OPT_EASYCRON='yes'` in package easycron.

DYNDNS_LOGINTIME If you use a provider FLI4L should log into regularly to prevent deactivation of the service (see above), you can use this variable to configure at what

time this login process is intended to take place. A time value in Cron format is required here, for details please read the Documentation of package easycron. The default setting is `0 8 * * *`, meaning a daily login at eight o'clock in the morning.

DYNDNS_ALLOW_SSL Setting this variable to `'yes'` will update over an encrypted SSL connection if possible.

DYNDNS_LOOKUP_NAMES The IP should only be updated if it really changed. Many fli4l routers don't have a permanent data storage like a harddisk where this information could be saved to be present at boot time. To prevent unnecessary updates flil4 may query name servers (only in this case) for its actually registrated IP. The information will be saved and used for further updates.

Note that after a reboot a new update interval will start if fli4l uses a name server to detect its IP.

DYNDNS_DEBUG_PROVIDER Setting this variable to `'yes'` will record a trace of the update process for debugging purposes. Default: `DYNDNS_DEBUG_PROVIDER='no'`

4.8. EASYCRON - Time-based Job Scheduling

This package was inititated by Stefan Manske email: fli4l@stephan.manske-net.de and adapted to version 2.1 by the fli4l-team.

4.8.1. Configuration

By using `OPT_EASYCRON` it is possible to execute commands at designated times by adding them to the config-file.

The following entries are supported:

OPT_EASYCRON `OPT_EASYCRON='yes'` activates the package

Default setting: `OPT_EASYCRON='no'`

EASYCRON_MAIL Avoid the sending of unwanted mails by crond.

Default setting: `EASYCRON_MAIL='no'`

EASYCRON_N The number of commands to be executed by cron.

EASYCRON_x_CUSTOM Those familiar with the settings in crontab may define additional settings for each entry (like MAILTO, PATH...). Entries have to be separated by \\. You should be really familiar with cron to use this option.

Default setting: `EASYCRON_CUSTOM=''`

EASYCRON_x_COMMAND Specify the command to be executed in `EASYCRON_x_COMMAND`, for example:

```
EASYCRON_1_COMMAND='echo 1 '>' /dev/console'
```

EASYCRON_x_TIME specifies execution times according to the cron syntax.

4.8.2. Examples

- The computer outputs “Happy new year!” to the console

```
EASYCRON_1_COMMAND = 'echo Happy new year! > /dev/console'
EASYCRON_1_TIME    = '0 0 31 12 *'
```

- xxx will be executed Monday to Friday from 7AM to 8PM Uhr every full hour.

```
EASYCRON_1_COMMAND = 'xxx'
EASYCRON_1_TIME    = '0 7-20 0 * 1-5'
```

- The router terminates the DSL internet connection each night at 03:40AM and reestablishes it after 5 seconds. Device names to be used: pppoe, ippp[1-9], ppp[1-9].

```
EASYCRON_1_COMMAND = 'fli4lctrl hangup pppoe; sleep 5; fli4lctrl dial pppoe'
EASYCRON_1_TIME    = '40 3 * * *'
```

Further informations on the cron syntax can be found here (german)

- http://www.pro-linux.de/artikel/2/146/der-batchdaemon-cron.html
- http://de.linwiki.org/wiki/Linuxfibel_-_System-Administration_-_Zeit_und_Steuerung#Die_Datei_crontab
- http://web.archive.org/web/20021229004331/http://www.linux-magazin.de/Artikel/ausgabe/1998/08/Cron/cron.html
- http://web.archive.org/web/20070810063838/http://www.newbie-net.de/anleitung_cron.html

4.8.3. Prerequisites

- fli4l version > 2.1.0
- for older versions please use opt_easycron version from the OPT-Database

4.8.4. Installation

Unzip `OPT_EASYCRON` to your fli4l directory like any other fli4l opt.

4.9. HD - Support For Harddisks, Flash-Cards, USB-Sticks, ...

4.9.1. OPT_HDINSTALL - Installation On A Harddisk/CompactFlash

fli4l supports a large variety of boot media (CD, HD, Network, Compact-Flash,...). Floppy Disks are not supported anymore due to filesize regressions.

All steps necessary to install to a harddisk are explained below.

The usual way is installation via a physical boot medium. Installing via network boot is possible too. `OPT_HDINSTALL` prepares the harddisk. If boot medium and installation target share the same `BOOT_TYPE='hd'` installation files will be transferred immedeately. If direct transfer is not possible the files will be transferred later via scp or remote update using Imonc.

An Overview to the different harddisk installation variants A or B is found at the Beginning of the documentation for fli4l (Page 14). Please read carefully before proceeding!

HD-Installation In Six Simple Steps

1. create a bootable fli4l medium with package `BASE` and `OPT_HDINSTALL`. This medium must be able to perform remote updates. Either `OPT_SSHD` must be activated or `START_IMOND` is set to `'yes'`. If additional drivers not contained in the standard installation are necessary to access the harddisk, configure `OPT_HDDRV` as well.

2. boot the router with the installation medium.

3. log in to the router console and execute "hdinstall.sh".

4. transfer the files syslinux.cfg, kernel, rootfs.img, opt.img and rc.cfg via scp or Imonc to the router's /boot directory if prompted to. It is recommended to work with two fli4l directories, one for the setup and a second for the hd installation. In the HD version, set the variable `BOOT_TYPE='hd'` and for the boot media type according to its type.

   During remote update the files for the hd-version have to be copied!

5. unload boot medium, shut down the router and reboot again (by using halt/reboot/poweroff). The router will boot from harddisk now.

6. if problems occur please refer to the following text.

HD-Installation Explained In Detail (including examples)

First, a router boot media containing the installation scripts and additional drivers (eventually) has to be created. Activate `OPT_HDINSTALL` in config/hd.txt and `OPT_HDDRV` (only if additional drivers are needed). Please read the section on `OPT_HDDRV` thoroughly!

The variable `BOOT_TYPE` in base.txt has to be set in accordance with the chosen setup media. After all, a setup has to be performed. . . The variable `MOUNT_BOOT` in base.txt has to set to `'rw'`, in order to allow saving new archives (*. img) loaded over the network.

Then boot the router from this setup medium. Run the installation program by executing "hdinstall.sh" at the fli4l sonsole . After answering a few questions the installion on the hard drive is performed. Eventually you will be prompted to load files needed for the router via remote update.

Don't forget this remote update, otherwise the router won't boot from harddisk. To reboot the routers after remote update use reboot/halt/poweroff, otherwise your remote update changes will be lost.

The installation script may be started directly at the router console or via ssh from another PC. This way you have to log in by giving a password. As an ssh client you may use the freeware 'putty'.

Configuration Of The Setup Bootmedia

At this point network configuration has to be completed in order to be able to copy files over the network later. Please do not activate DNS_DHCP at this point because this may cause all

BOOT_TYPE	set according to type of bootmedia for the installation
MOUNT_BOOT='rw'	necessary to copy new archives (*.img) to the harddisk over network
OPT_HDINSTALL='yes'	necessary to have the setup script and tools for formatting of partitions on the bootmedia
(OPT_HDDRV='yes')	only necessary if harddisk needs special drivers
OPT_SSHD='yes'	after preparation of the harddisk eventually files have to be copied via remote update. You will either need sshd, imond (IMOND='yes') or another package allowing file transfers.

Table 4.6.: Configuration example of a setup media

kinds of errors (the DHCP-server maybe already have a lease file for the router to be installed). For a remote update via scp (package SSHD) please set up OPT_SSHD='yes'. As an alternative you may transfer files via IMOND. This needs a complete and working configuration for DSL or ISDN. Please omit all packages not mentioned, i.e. no DNS_DHCP, SAMBA_LPD, LCD,Portforwarding a.s.o.

In case that the installations stops with the error message

```
*** ERROR: can't create new partition table, see docu ***
```

several problems may have occurred:

- harddisk is in use, maybe by an installation canceled before. Reboot and try again.
- additional drivers are needed, see OPT_HDDRV
- hardware problems, see appendix.

In the last step the final configuration files can be set up and all other packages needed may be added to the router.

Examples For Completed Installations Type A and B:

An example for each configuration is to be found in table 4.7.

BOOT_TYPE='hd'	necessary because we are booting from harddisk now
MOUNT_BOOT='rw|ro|no'	choose one. For copying new fli4l archives to harddisk over network 'rw' is needed.
OPT_HDINSTALL='no'	not needed after successful installation.
OPT_MOUNT	activate only if a data partition was configured
(OPT_HDDRV='yes')	only necessary if harddisk can't be used without additional drivers.

Table 4.7.: Example for an installation according to type A or B

Creation of a swap-partition will only be available if the router has less than 32MB RAM and the installation target is NO flash media!

4.9.2. OPT_MOUNT - Automatic Mounting Of Filesystems

OPT_MOUNT mounts data partitions created during installation to /data, file system checks will be performed automatically when needed. CD-ROMs will be mounted to /cdrom if a CD is inserted. For swap-partitions OPT_MOUNT is not needed!

OPT_MOUNT reads the configuration file hd.cfg on the boot-partition and mounts partitions mentioned there. If `OPT_MOUNT` was transferred via remote update to an already installed router this file has to be edited manually.

While booting from CD-ROM OPT_MOUNT can't be used. The CD may be mounted by setting `MOUNT_BOOT='ro'`.

The file hd.cfg on the DOS-partition for a router according to type B with swap and data partition looks like this (example):

```
hd_boot='sda1'
hd_opt='sda2'
hd_swap='sda3'
hd_data='sda4'
hd_boot_uuid='4A32-0C15'
hd_opt_uuid='c1e2bfa4-3841-4d25-ae0d-f8e40a84534d'
hd_swap_uuid='5f75874c-a82a-6294-c695-d301c3902844'
hd_data_uuid='278a5d12-651b-41ad-a8e7-97ccbc00e38f'
```

Just omit non-existent partitions, a router according to type A without further partitions looks like this:

```
hd_boot='sda1'
hd_boot_uuid='4863-65EF'
```

4.9.3. OPT_EXTMOUNT - Manual Mounting Of File Systems

OPT_EXTMOUNT mounts data partitions to any chosen mountpoint in file system. This allows to mount file systems created manually and for example provide a rsync-server directory.

EXTMOUNT_N Number of manually created data partitions to be mounted.

EXTMOUNT_x_VOLUMEID Device, label or UUID of the volume to be mounted. By executing 'blkid' device, label and UUID of all volumes can be displayed.

EXTMOUNT_x_FILESYSTEM The file system used for the partition. fli4l supports isofs, fat, vfat, ext2, ext3 und ext4 at the time of writing.
(The default setting `EXTMOUNT_x_FILESYSTEM='auto'` automatically tries to determine the file system used.)

EXTMOUNT_x_MOUNTPOINT The path (Mountpoint) to where the device should be mounted. It does not have to exist and will be created automatically.

EXTMOUNT_x_OPTIONS Specify special options to be passed to the 'mount' command here.

Example:

```
EXTMOUNT_1_VOLUMEID='sda2'        # device
EXTMOUNT_1_FILESYSTEM='ext3'      # filesystem
EXTMOUNT_1_MOUNTPOINT='/mnt/data' # mountpoint for device
EXTMOUNT_1_OPTIONS=''             # extra mount options passed via mount -o
```

4.9.4. OPT_HDSLEEP – Setting Automatic Sleep Mode For Harddisks

A harddisk can power down after a certain time period without activity. The disk will save power and operate quiet then. Accessing the harddisk will cause it to automatically spin up again.

Not all harddisks tolerate frequent spinup. Don't set the time for spindown too short. Older IDE-disks don't even have this function. This setting is also senseless for Flash-Media.

HDSLEEP_TIMEOUT The variable specifies after what time period without access the disk should power down. It will power down after that time period and come up again with the next access. Sleep timeouts can be specified in minutes from one to 20 and in periods of 30 minutes from half an hour up to 5 hours. A sleep timeout of 21 or 25 minutes will be rounded to 30 minutes. Some harddisks ignore values too high and stop after some minutes then. Please test the settings thoroughly because proper functioning depends on the hardware used!

```
HDSLEEP_TIMEOUT='2'             # wait 2 minutes until power down
```

4.9.5. OPT_RECOVER – Emergency Option

This variable determines if functions for creating an emergency option will be available. If activated the option copies the command "mkrecover.sh" to the router. By executing it you can activate the emergency option at the console. With package "HTTPD" installed the action of copying an existing installation to an emergency instance can be achieved conveniantly in the menu "recover".

To use the recovery installation choose "r" for recovery in the boot menu at the next reboot.

```
OPT_RECOVER='yes'
```

4.9.6. OPT_HDDRV - Additional Drivers For Harddisk Controllers

By setting `OPT_HDDRV`='yes' you may activate drivers additionally needed. Generally this is NOT needed for IDE und SATA, package 'Base' will load all necessary files.

HDDRV_N Set the number of drivers to be loaded.

HDDRV_x `HDDRV_1` a.s.o. Set drivers to be chosen for the host-adapters in use. A list of all supported adapters is provided with the initial config file for package hd.

HDDRV_x_OPTION With HDDRV_x_OPTION additional options can be passed that some drivers need for proper operation (for example an IO-address). This variable can be empty for the most drivers.

In the Appendix (Page 350) you may find an overview of the most common errors concerning harddisk and CompactFlash operation.

Example 1: Access to a SCSI-harddisk on an Adaptec 2940 controller

```
OPT_HDDRV='yes'             # install Drivers for Harddisk: yes or no
HDDRV_N='1'                 # number of HD drivers
HDDRV_1='aic7xxx'           # various aic7xxx based Adaptec SCSI
HDDRV_1_OPTION=''           # no need for options yet
```

Example 2: Accelerated IDE-Access with PC-Engines ALIX

```
OPT_HDDRV='yes'             # install Drivers for Harddisk: yes or no
HDDRV_N='1'                 # number of HD drivers
HDDRV_1='pata_amd'          # AMD PCI IDE/ATA driver (e.g. ALIX)
HDDRV_1_OPTION=''           # no need for options yet
```

4.10. HTTPD - Webserver For Status-Display

4.10.1. OPT_HTTPD - Mini-Webserver As Status-Display

The webserver can be used to display or change the status of fli4l routers (IMONC can be used too). The status monitor can be activated by setting OPT_HTTPD='yes'.

If you are using the default configuration set your browser to one of the following addresses:

```
http://fli4l/
http://fli4l.domain.lan/
http://192.168.6.1/
```

If you configured fli4l to use another name and/or domain these have to be used. If the webserver is set to listen on another port than the default one specify it like this:

```
http://fli4l:81/
```

As of version 2.1.12 a login page will be displayed which is not protected by a password. Protected pages are in the subdirectory admin, for example:

```
http://fli4l.domain.lan/admin/
```

The web server can be configured by setting the following variables:

HTTPD_GUI_LANG This specifies the language in which the web interface is shown. If set to 'auto' the language setting is taken from the variable LOCALE in base.txt.

HTTPD_GUI_SKIN This Variable affects the appearance of the surface. Package *httpd_skins* provides two different values *fixed* and *notfixed*. Default is *default*.

HTTPD_LISTENIP The web server usually binds to a so-called wildcard address in order to be accessed on any router interface. Set the web server with this parameter to only bind to one IP address. The corresponding IP address is given here: `IP_NET_x_IPADDR`. Normally this parameter is left blank, so the default (Accessible on any interface IP) is used.

This parameter is used to bind the httpd to only one IP so that other instances can bind to other IPs on the router. It can not be used to limit access to the web interface of the router. This would need additional configuring of the packet filter, too.

It is also possible to specify multiple IP addresses here (separated by spaces).

HTTPD_PORT Set this value if the web server should run on another port than 80. This is usually not recommended, since then it must be accessed through the browser by adding the port number. Example http://fli4l:81/.

HTTPD_PORTFW If setting this variable to 'yes' you can change port forwarding via the web interface. Rules can be erased and/or added. Changes take effect immediately and only apply during router runtime. If the router is restarted, the changes are gone. The default setting is 'yes'.

HTTPD_ARPING The web server displays the online/offline state of the hosts listed in `HOST_x`. To achieve this it uses the *"Arp-Cache"*, a cache that buffers the addresses of the local hosts. If a host does not communicate with the router for longer its address will disappear from the *"Arp-Cache"* and it appears to be offline. If you want to keep the *"Arp-Cache"* up-to-date (keep the online hosts that are not communicating with the router in it) set `HTTPD_ARPING='yes'?`.

HTTPD_ARPING_IGNORE_N Sets the number of entries to be ignored when arping

HTTPD_ARPING_IGNORE_x IP-adress or name of the hosts not to be included in ARPING-tests. This may be useful for battery based hosts consuming too much power by regular network queries (i.e. laptops or cell phones in WLAN).

4.10.2. User Management

The webserver provides a sophisticated user management:

HTTPD_USER_N Specify the number of users. If set to 0 user management will be switched off completely so everybody can access the web server.

HTTPD_USER_x_USERNAME HTTPD_USER_x_PASSWORD HTTPD_USER_x_RIGHTS enter username and password for each user here. On top specify for each user which functions of the the web server should be accessible to him. Functions are controlled via the variable `HTTPD_RIGHTS_x`. In the simplest case it is 'all', which means that the corresponding user is allowed to access everything. The variable has the following structure:

```
'Range1: right1,right2,... Range2:...'
```

Instead of adding all rights for a certain range the word "all" can be used. This means that the user has all rights in this range. The following ranges and rights exist:

Range "status" Everything in menu 'Status'.

view User can access all menu items.

dial User can dial and hang up connections.

boot User can reboot and shut down the router.

link User can switch channel bundeling.

circuit User can switch circuits.

dialmode User can switch dialmodes (Auto, Manual, Off).

conntrack User can view currently active connections.

dyndns User can view logfiles of package DYNDNS (Page 110).

Range "logs" Everything concerning log files (connections, calls, syslog)

view User can view logfiles.

reset User can delete logfiles.

Range "support" Everything of use for getting help (Newsgroups a.s.o.).

view User can access links to documentation, fli4l-website, a.s.o.

systeminfo User can query detailed informations about configuration and actual status of the router (i.e.: firewall).

Some examples:

HTTPD_USER_1_RIGHTS='all' Allow a user to access everything in the webserver!

HTTPD_USER_2_RIGHTS='status:view logs:view support:all' User can view everything but can't change settings.

HTTPD_USER_3_RIGHTS='status:view,dial,link' User can view the status of internet connections, dial and switch channel bundeling.

HTTPD_USER_4_RIGHTS='status:all' User can do everything concerning internet connections and reboot/shutdown. He is not allowed to view or delete logfiles (nor timetables).

4.10.3. OPT_OAC - Online Access Control

OPT_OAC (optional)

Activates 'Online Access Control'. By using this internet access of each host configured in package dns_dhcp (Page 91) can be controlled selectively.

A console tool is available too, providing an interface to other packages like EasyCron:

/usr/local/bin/oac.sh

Options will be shown when executed on a console.

OAC_WANDEVICE (optional)

Restricts the online access control to connections on this network device (i.e. 'Pppoe').

OAC_INPUT (optional)

Provides protection against circumvention via proxy.

OAC_INPUT='default' blocks default ports for Privoxy, Squid, Tor, SS5, Transproxy.

OAC_INPUT='tcp:8080 tcp:3128' blocks TCP Port 8080 and 3128. This is a space separated list of ports to be blocked and their respective protocol (udp, tcp). Omitting protocols blocks both udp and tcp.

Omitting this variable or setting it to `'no'` deactivates the function.

OAC_ALL_INVISIBLE (optional)

Turns off overview if at least one group exists. If no groups exist the variable is without effect.

OAC_LIMITS (optional)

List of available time limits separated by spaces. Limits are set in minutes. This allows time period based blocking or access definition.

Default: '30 60 90 120 180 360 540'

OAC_MODE (optional)

Possible values: `'DROP'` or `'REJECT'` (default)

OAC_GROUP_N (optional)

Number of client groups. Used for clarity but also allows to block or allow access for a whole group at once over the web interface.

OAC_GROUP_x_NAME (optional)

Name of the group - this name will be displayed in the web interface and may be used in console script 'oac.sh'.

OAC_GROUP_x_BOOTBLOCK (optional)

If set to `'yes'` all clients of the group are blocked at boot. Useful if PCs should be blocked in general.

OAC_GROUP_x_INVISIBLE (optional)

Marks the group as invisible. Useful to block a PC in general which should not be visible in the web interface. The console script oac.sh is not affected by this (for use in easycron).

OAC_GROUP_x_CLIENT_N (optional)

Number of clients in the group.

OAC_GROUP_x_CLIENT_x (optional)

Name of the client - See package dns_dhcp (Page 91).

OAC_BLOCK_UNKNOWN_IF_x (optional)

List of interfaces defined in base.txt allowing internet access only to hosts defined in dns_dhcp.txt. Hosts not defined are blocked in general.

4.11. HWSUPP - Hardware support

4.11.1. Description

This package supplies the support for special hardware components.
Supported are:

- Temperature sensors
- LEDs
- Voltage sensors
- Fan speed
- Buttons
- Watchdog
- VPN cards

The following systems, mainboards and VPN cards are supported:

- Standard PC hardware
  - PC keyboard LEDs
- ACPI Hardware
- Embedded systems
  - AEWIN SCB6971
  - Fujitsu Siemens Futro S200
  - PC Engines ALIX
  - PC Engines APU
  - PC Engines WRAP
  - Soekris net4801
  - Soekris net5501
- Mainboards
  - Commell LE-575
  - GigaByte GA-M521-S3
  - LEX CV860A
  - SuperMicro PDSME
  - SuperMicro X7SLA
  - Tyan S5112
  - WinNet PC640
  - WinNet PC680
- VPN cards (PCI, miniPCI and miniPCIe)
  - vpn1401 vpn1411

4.11.2. Configuration of the HWSUPP package

The configuration is made, as for all fli4l packages, by adjusting the file `path/fli4l-3.10.5/<config>/hwsupp.txt` to meet your own demands.

OPT_HWSUPP The setting `'no'` deactivates the OPT_HWSUPP package completely. There will be no changes made to the fli4l boot medium or the archive `opt.img`. OPT_HWSUPP does not overwrite any other parts of the fli4l installation. To activate OPT_HWSUPP set the variable `OPT_HWSUPP` to `'yes'`.

HWSUPP_TYPE This configuration variable sets the type of supported hardware. Following values can be used:

- sim
- generic-pc
- generic-acpi
- aewin-scb6971
- commell-le575
- fsc-futro-s200
- gigabyte-ga-m52l-s3
- lex-cv860a
- pcengines-alix
- pcengines-apu
- pcengines-wrap
- soekris-net4801
- soekris-net5501
- supermicro-pdsme
- supermicro-x7sla
- tyan-s5112
- winnet-pc640
- winnet-pc680

HWSUPP_WATCHDOG The setting `'yes'` activates the watchdog daemon if the hardware contains a watchdog. The watchdog will automatically restart a non responding system.

HWSUPP_CPUFREQ The setting `'yes'` activates CPU frequency adjustment controls.

HWSUPP_CPUFREQ_GOVERNOR Selection of CPU frequency governor. The selected governor controls the frequency adjustment behaviour. It's a selection of one of:

- performance
  The CPU allways runs with the highest available frequency.
- ondemand
  The CPU frequency will be adjusted depending on the current CPU usage. The frequency can change very quickly.

- conservative
  The CPU frequency will be adjusted depending on the current CPU usage. The frequency is changed step by step.
- powersave
  The CPU allways runs with the lowest available frequency.
- userspace
  The CPU frequence kann be set manually or by an user script via the sysfs variable `/devices/system/cpu/cpu<n>/cpufreq/scaling_setspeed`.

HWSUPP_LED_N Defines the number of LEDs. The number of LEDs of the hardware in use should be entered here.

HWSUPP_LED_x Defines the information indicated by the LED. The following informations are possible:

- ready - the fli4l router ist ready for operation[6]
- online - the fli4l router has an active internet connection
- trigger - LED is controlled by a kernel trigger
- user - LED is controlled by an user script

The list of possible indications can be extended by other packages. For example, if the WLAN package is loaded the information

- wlan - WLAN is active

is possible.

In apppendix B.10 package developers can get some hints on how to create such extensions.

HWSUPP_LED_x_DEVICE Specifies the LED device.

Here you either have to enter a LED device (to be found at `/sys/class/leds/` in the router's file system) or a GPIO[7]number.

A list of valid LED device names for a specific `HSUPP_TYPE` can be found in the appendix B.7.1.

The GPIO number has to be enterd in the format `gpio::x`. If a GPIO is entered, the corresponding LED device will be created automatically. By preceding the char `/` the GPIO functionality may be inverted.

Examples:

```
HWSUPP_LED_1_DEVICE='apu::1'      # LED 1 on PC engines APU
HWSUPP_LED_2_DEVICE='gpio::237'   # GPIO 237
HWSUPP_LED_3_DEVICE='/gpio::245'  # inverted GPIO 245
```

[6] If `HWSUPP_LED_x='ready'` is set, the boot progress is indicated by a blink sequence (see appendix B.9).
[7] A GPIO (General Purpose Input/Output) is a generic pin on an integrated circuit whose behavior can be programmed at run time, including whether it is an input or output pin.

HWSUPP_LED_x_PARAM Defines parameters for the selected LED information.

Depending on the selection in in `HWSUPP_LED_x`, in `HWSUPP_LED_x_PARAM` different settings are possible.

If `HWSUPP_LED_x='trigger'` is set, the trigger name has to be specified in `HWSUPP_LED_x_PARAM`.

Available triggers can be displayed with the shell command `cat /sys/class/leds/*/trigger`.

Besides triggers created by e.g. netfilter or hardware drivers like ath9k, further trigger modules can be loaded via `HWSUPP_DRIVER_x`.

Examples:

```
HWSUPP_LED_1='trigger'
HWSUPP_LED_1_TRIGGER='heartbeat'
HWSUPP_LED_2='trigger'
HWSUPP_LED_2_TRIGGER='netfilter-ssh'
```

If `'HWSUPP_LED_x'` has the value `'user'` in `HWSUPP_LED_PARAM` a valid script name including path has to be entered.

Example:

```
HWSUPP_LED_1='user'
HWSUPP_LED_1_PARAM='/usr/local/bin/myledscript'
```

When `HWSUPP_LED_x='wlan'` is set, the WLAN devices have to be entered in `HWSUPP_LED_x_PARAM`.

Defines one ore more WLAN devices, whose state shall be displayed. Multiple WLAN devices have to be separated by spaces.

When the state of multiple WLAN devices should be indicated by a single LED, the LED has the following meaning:

- off - all WLAN devices are inactiv
- blinking - some WLAN device(s) is/are active
- on - all WLAN devices are active

Example:

```
HWSUPP_LED_1='wlan'
HWSUPP_LED_1_WLAN='wlan0 wlan1'
```

HWSUPP_BOOT_LED Definies a LED to indicate the boot progress by a blink sequence.

When `HWSUPP_LED_x='ready'` is set for any LED, this setting is used and `HWSUPP_BOOT_LED` will be ignored.

HWSUPP_BUTTON_N Defines the number of buttons.
The number of buttons of the hardware in use should be entered here.

HWSUPP_BUTTON_x Defines the action which should be executed on button press. The following actions are supported:

- reset - restart the fli4l router
- online - causes an internet dialin or terminates an internet connection.
- user - an user script will be executed

Th e list of possible actions can be extended by other packages. If the WLAN package is loaded, eg. the action

- wlan - activate or deactivate WLAN

is possible.

HWSUPP_BUTTON_x_DEVICE Specifies the button device an.

Here has to be entered a GPIO number in the format `gpio::x`. By preceding the char `/` the GPIO functionality may be inverted.

A list of predefined GPIO's for a specific `HSUPP_TYPE` can be found in the appendix B.7.2.

Examples:

```
HWSUPP_BUTTON_1_DEVICE='gpio::252'
HWSUPP_BUTTON_2_DEVICE='/gpio::237'
```

HWSUPP_BUTTON_x_PARAM Defines parameters for the action selected in `HWSUPP_BUTTON_x`.

Depending on the action `HWSUPP_BUTTON_x_PARAM` has different meanings.

If `HWSUPP_BUTTON_x='user'` is set, `HWSUPP_BUTTON_x_PARAM` defines a script to be executed on button press.

Example:

```
HWSUPP_BUTTON_1='user'
HWSUPP_BUTTON_2_WLAN='/usr/local/bin/myscript'
```

If `HWSUPP_BUTTON_x` is set to `'wlan'`, the `HWSUPP_BUTTON_x_PARAM` defines one ore more WLAN devices, which shall be activated or deactivated on button press. Multiple WLAN devices have to be separated by spaces.

Example:

```
HWSUPP_BUTTON_2='wlan'
HWSUPP_BUTTON_2_WLAN='wlan0 wlan1'
```

4.11.3. Expert settings

The following settings should only be touched if you know exactly

- which hardware you have,
- which additional drivers it needs and
- the addresses and types of I²C[8]devices.

Activating the expert settings will issue a warning during the mkfli4l build.

HWSUPP_DRIVER_N Number of additional drivers. The drivers in `HWSUPP_DRIVER_x` will be loaded in the denoted order.

HWSUPP_DRIVER_x Driver name (without file extension `.ko`).

Example:

```
HWSUPP_DRIVER_N='2'
HWSUPP_DRIVER_1='i2c-piix4'     # I2C bus driver
HWSUPP_DRIVER_2='gpio-pcf857x'  # I2C GPIO expander
```

HWSUPP_I2C_N Number of I²C devices to be loaded.

I²C doesn't support any PnP mechanismn. Hence for each I²C device the bus number, the device address and the device type have to be specified.

HWSUPP_I2C_x_BUS I²C bus number the device is attached to.

The bus number has to be entered as `i2c-x`.

HWSUPP_I2C_x_ADDRESS The device's I²C address.

The address has to be entered as a hex number in the range between `0x03` and `0x77`.

HWSUPP_I2C_x_DEVICE The type of I²C device which is supported by an already loaded driver.

Example:

```
HWSUPP_I2C_N='1'
HWSUPP_I2C_1_BUS='i2c-1'
HWSUPP_I2C_1_ADDRESS='0x38'
HWSUPP_I2C_1_DEVICE='pcf8574a' # supported by gpio-pcf857x driver
```

4.11.4. Support for VPN cards

OPT_VPN_CARD The setting `'no'` deactivates the OPT_VPN_CARD package completely. There will be no changes made to the fli4l boot mediums or the archive `opt.img`. OPT_VPN_CARD does not overwrite any other parts of the fli4l installation. To activate OPT_VPN_CARD set the variable `OPT_VPN_CARD` to `'yes'`.

[8] An I²C bus or SMBus is a serial bus used in PCs eg. to read temperature sensor values. In many cases an I²C bus or SMBus is available on a pin header and can be used for own hardware extensions.

VPN_CARD_TYPE This configuration variable defines the type of the VPN accelerator. The following values are supported:

- hifn7751 - Soekris vpn1401 and vpn1411
- hifnhipp

4.12. IPv6 - Internet Protocol Version 6

4.12.1. 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, [9] 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. [10]

4.12.2. 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`

[9] only nearly because some addresses are used for specialized purposes, such as for broadcast and multicasting

[10] 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.

4.12.3. 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 explicitely. 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)*

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*

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 works 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 informations 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 no support IPv6 natively. With an internet host of 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). [11] 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'`

[11] 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 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='passwort'`

IPV6_TUNNEL_x_TUNNELID This variable contains the identificator 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. [12] 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.

[12] 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 Procotol) 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 revceived 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='tmpl: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-dubnets 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).

- 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.

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.

4.12.4. Web-GUI

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

4.13. ISDN - Communication Over Active And Passive ISDN-Cards

fli4l is mainly aiming to be used as an ISDN- and/or DSL-router. By setting OPT_ISDN='yes' the ISDN package is activated. Precondition is a ISDN-card supported by fli4l.

Default setting: OPT_ISDN='no'

4.13.1. Establishing An ISDN Connection

fli4l's behaviour during dial-in is determined by three variables DIALMODE, ISDN_CIRC_X_ROUTE_X, ISDN_CIRC_X_TIMES. DIALMODE (Page 70) (in <config>/base.txt) determines wether a connection will be automatically established on an active circuit on packet arrival or not. DIALMODE may have the following values:

auto If a packet reaches an ISDN-circuit (res. the ISDN interface derived from it - ippp*) a connection will be established automatically. If and when a packet reaches an ISDN-circuit is determined by ISDN_CIRC_X_ROUTE_X and ISDN_CIRC_X_TIMES.

manual In manual mode the connection has to established via imond/imonc. How this is done see imonc/imond.

off No ISDN connections will be established.

Which circuits packets will trigger a dial-in is defined by ISDN_CIRC_X_ROUTE_X. Normally this uses '0.0.0.0/0' as the 'default route'. This means that all packets that leave the local net are using this circuit if it is active. If and when it is active is determined by ISDN_CIRC_X_TIMES for fli4l is doing *least cost routing* over a predefinded circuit (see Least-Cost-Routing - Functionality (Page 263) in the base documentation). If not all but only packets for a certain net should be routed over this circuit (i.e. a company net) additional nets can be given here. fli4l will then set a permanently active ISDN route over the interface set for this circuit. If a packet is sent to this net a connection will be automatically established.

As said before ISDN_CIRC_X_TIMES besides the connection costs for a circuit describes also if and when a circuit with a default route is active and can trigger a connection. 'When' is defined by the time-info, the first two elements (i.e. Mo-Fr:09-18), 'if' is given by the forth

parameter lc-default-route (y/n). fli4l (res. imond) will trigger a connection to the internet provider and assure that all packets leaving the local net are routed over the circuit that is active at this time.

The standard use cases in summary:

- If simply a connection to the internet is intended `DIALMODE` is set to auto, 1-n circuits are to be defined which have an initial route of '0.0.0.0/0' and whose times (times with lc-default-route = y) cover the whole week.

```
ISDN_CIRC_%_ROUTE_N='1'
ISDN_CIRC_%_ROUTE_1='0.0.0.0/0'
ISDN_CIRC_%_TIMES='Mo-Su:00-24:0.0148:Y'
```

- If besides that a connection to a company net should be used another (or more) circuits have to be defined. The route should differ from '0.0.0.0/0' to accomplish a permanently active route.

```
ISDN_CIRC_%_ROUTE_N='1'
ISDN_CIRC_%_ROUTE_1='network/netmaskbits'
ISDN_CIRC_%_TIMES='Mo-Su:00-24:0.0148:Y'
```

4.13.2. ISDN Card

ISDN_TYPE ISDN_IO ISDN_IO0 ISDN_IO1 ISDN_MEM ISDN_IRQ ISDN_IP ISDN_PORT

Some technical data about the ISDN card is specified here.

The values in the example work for a TELES 16.3 set to IO-address 0xd80 via Dip-switches. For other settings of the switches the values have to be changed.

Common error (example):

```
ISDN_IO='240' -- right value would be: ISDN_IO='0x240'
```

Using IRQ 12 eventually a PS/2 mouse has to be deactivated in BIOS. Better choose another IRQ! „Good ones“ are mostly 5, 10 and 11.

`ISDN_TYPE` in principle follows the type numbers for HiSax drivers. Exception: non-HiSax-cards like i.e. the AVM-B1. For those the type numbers were extended (see below). The list of possible HiSax-types is based on
`linux-2.x.y/Documentation/isdn/README.HiSax`.

Typ	Karte	Needed parameters
Dummy Type-Number:		
0	no driver (dummy)	none
Typen-Numbers for HiSax drivers:		
1	Teles 16.0	irq, mem, io
2	Teles 8.0	irq, mem
3	Teles 16.3 (non PnP)	irq, io
4	Creatix/Teles PnP	irq, io0 (ISAC), io1 (HSCX)
5	AVM A1 (Fritz)	irq, io
5	AVM (Fritz!Card Classic)	irq, io

Typ	Karte	Needed parameters
6	ELSA PCC/PCF cards	io or nothing for autodetect (the iobase is required only if you have more than one ELSA card in your PC)
7	ELSA Quickstep 1000	irq, io (from isapnp setup)
8	Teles 16.3 PCMCIA	irq, io
9	ITK ix1-micro Rev.2	irq, io (from isapnp setup?)
10	ELSA PCMCIA	irq, io (set with card manager)
11	Eicon.Diehl Diva ISA PnP	irq, io
11	Eicon.Diehl Diva PCI	no parameter
12	ASUS COM ISDNLink	irq, io (from isapnp setup)
13	HFC-2BS0 based cards	irq, io
14	Teles 16.3c PnP	irq, io
15	Sedlbauer Speed Card	irq, io
15	Sedlbauer PC/104	irq, io
15	Sedlbauer Speed PCI	no parameter
16	USR Sportster internal	irq, io
17	MIC card	irq, io
18	ELSA Quickstep 1000PCI	no parameter
19	Compaq ISDN S0 ISA card	irq, io0, io1, io (from isapnp setup io=IO2)
20	NETjet PCI card	no parameter
21	Teles PCI	no parameter
22	Sedlbauer Speed Star (PCMCIA)	irq, io (set with card manager)
23	reserved (AMD 7930)	n.a.
24	Dr. Neuhaus Niccy PnP	irq, io0, io1 (from isapnp setup)
24	Dr. Neuhaus Niccy PCI	no parameter
25	Teles S0Box	irq, io (of the used lpt port)
26	AVM A1 PCMCIA (Fritz!)	irq, io (set with card manager)
27	AVM PnP (Fritz!PnP)	irq, io (from isapnp setup)
27	AVM PCI (Fritz!PCI)	no parameter
28	Sedlbauer Speed Fax+	irq, io (from isapnp setup)
29	Siemens I-Surf 1.0	irq, io, memory (from isapnp setup)
30	ACER P10	irq, io (from isapnp setup)
31	HST Saphir	irq, io
32	Telekom A4T	none
33	Scitel Quadro	subcontroller (4*S0, subctrl 1...4)
34	Gazel ISDN cards (ISA)	irq,io
34	Gazel ISDN cards (PCI)	none
35	HFC 2BDS0 PCI	none
36	W6692 based PCI cards	none
37	2BDS0 S+, SP	irq,io
38	NETspider U PCI	none
39	2BDS0 SP/PCMCIA [13]	irq,io (set with card manager)
40	not used (hotplug)	n.a.
41	Formula-n enter:now PCI	none
81	ST5481 USB ISDN adapters	none
82	HFC USB based ISDN adapters	none
83	HFC-4S/8S based ISDN cards	none
84	AVM Fritz!Card PCI/PCIv2/PnP	none

[13] for older versions type 84

Typ	Karte	Needed parameters
Type-numbers for Capi-drivers:		
100	Generic CAPI device without ISDN functionality, i.e. AVM Fritz!DSL SL	no parameter
101	AVM-B1 PCI	no parameter
102	AVM-B1 ISA	irq, io
103	AVM-B1/M1/M2 PCMCIA	no parameter
104	AVM Fritz!DSL	no parameter
105	AVM Fritz!PCI	no parameter
106	AVM Fritz!PNP	irq, io (from isapnp setup)
107	AVM Fritz!Classic	irq, io
108	AVM Fritz!DSLv2	no parameter
109	AVM Fritz!USBv2	no parameter
110	AVM Fritz!DSL USB	no parameter
111	AVM Fritz!USB	no parameter
112	AVM Fritz!X USB	no parameter
113	AVM FRITZ!DSL USBv2	no parameter
114	AVM FRITZ!PCMCIA	no parameter
160	AVM Fritz!Box Remote-Capi	ip,port
161	Melware Remote CAPI (rcapi)	ip,port
Type-Numbers for other drivers:		
201	ICN 2B	io, mem
Typen-Numbers for mISDN-drivers (experimental):		
301	HFC-4S/8S/E1 multiport cards	no parameter
302	HFC-PCI based cards	no parameter
303	HFCS-USB Adapters	no parameter
304	AVM FritZ!Card PCI (v1 and v2) cards	no parameter
305	cards based on Infineon (former Siemens) chips: - Dialogic Diva 2.0 - Dialogic Diva 2.0U - Dialogic Diva 2.01 - Dialogic Diva 2.02 - Sedlbauer Speedwin - HST Saphir3 - Develo (former ELSA) Microlink PCI (Quickstep 1000) - Develo (former ELSA) Quickstep 3000 - Berkom Scitel BRIX Quadro - Dr.Neuhaus (Sagem) Niccy	no parameter
306	NetJet TJ 300 and TJ320 cards	no parameter
307	Sedlbauer Speedfax+ cards	no parameter
308	Winbond 6692 based cards	no parameter

My card is a Teles 16.3 NON-PNP ISA, this is Type=3.

For a ICN-2B-card IO and MEM have to be set, for example `ISDN_IO`='0x320', `ISDN_MEM`='0xd0000'.

For newer Teles-PCI-card type=20 (instead of 21) has to be used. Those are shown by

"cat /proc/pci" as "tiger" or similar.

To use ISDN types 104 to 114 the matching drivers have to be downloaded from http://www.fli4l.de/download/stabile-version/avm-treiber/. Unpack them to the fli4l directory. They cannot be included because these drivers are not gpl'd.

For ISDN types 81, 82, 109 to 113 and 303 it is necessary to activate USB support. See USB - Support for USB-devices (Page 230).

To use ISDN types 10, 22, 26, 39, 103 or 114 it is necessary to activate PCMCIA PC-Card support. See PCMCIA - PC-Card Support (Page 183).

If you really don't know what card is in your PC you can get tips for type numbers also from the i4l-FAQ or mailing list.

Card types that are signed „from isapnp setup" have to be initialized by the PnP tool isapnp - if they really are PnP cards. See OPT_PNP - Installation of isapnp tools (Page 74).

ISDN type 0 is used if the ISDN package should be installed without an ISDN card, for example to use imond in a network router.

ISDN_DEBUG_LEVEL Sets the debug level for the HiSaX driver. Debug level is concatenated by addition of the following values (cited from the orginal docs):

Number	Debug-Information
1	Link-level <–> hardware-level communication
2	Top state machine
4	D-Channel Q.931 (call control messages)
8	D-Channel Q.921
16	B-Channel X.75
32	D-Channel l2
64	B-Channel l2
128	D-Channel link state debugging
256	B-Channel link state debugging
512	TEI debug
1024	LOCK debug in callc.c
2048	More debug in callc.c (not for normal use)

The default setting (ISDN_DEBUG_LEVEL='31') should be enough for most purposes.

ISDN_VERBOSE_LEVEL Sets the "verbosity" of the ISDN subsystem in fli4l kernel. Each verbose-level includes levels with lower numbers. Verbose levels are:

'0'	no additional informations
'1'	events triggering an ISDN connection will be logged
'2' and '3'	Calls are logged
'4' and more	Data transfer rates will be logged

Messages are sent over the kernel logging interface activated by OPT_SYSLOGD (Page 71).

Important: *If calls should be logged with telmond don't set this value lower than 2 otherwise telmond would lack informations for logging.*

Default setting: `ISDN_VERBOSE_LEVEL='2'`

ISDN_FILTER Activates filtering mechanism of the kernel to achieve hangup after the specified hangup timeout. See http://www.fli4l.de/hilfe/howtos/basteleien/hangup-problem-loesen/ for additional informations.

ISDN_FILTER_EXPR Specifies the filter to use if `ISDN_FILTER` is set to 'yes'.

4.13.3. OPT_ISDN_COMP (EXPERIMENTAL)

`OPT_ISDN_COMP='yes'` activates LZS- and BSD-compression. Credits for this go to Arwin Vosselman (email: arwin(at)xs4all(dot)nl). This addon package is in experimental state.

Default setting: `OPT_ISDN_COMP='no'`

The needed parameters for LZS-compression in detail:

ISDN_LZS_DEBUG (EXPERIMENTAL) Debug-level-settings:

- '0' no debugging informations
- '1' normal debugging informations
- '2' enhanced debugging informations
- '3' extreme debugging informations (incl. dumping of data packets)

Default setting: `ISDN_LZS_DEBUG='1'`

ISDN_LZS_COMP (EXPERIMENTAL) Compression level (not decompression!). Please use value 8. Values from 0 to 9 are possible.

Higher numbers will compress better at the cost of higher CPU load with 9 being disproportional excessive.

Default setting: `ISDN_LZS_COMP='8'`

ISDN_LZS_TWEAK (EXPERIMENTAL) Keep this value at '7' at the moment.

Default setting: `ISDN_LZS_TWEAK='7'`

Beside this three values the variable `ISDN_CIRC_x_FRAMECOMP` has to be set (see next chapter).

4.13.4. ISDN-Circuits

More connections over ISDN can be defined in fli4l configuration. A maximum of two at a time is possible over one ISDN card.

Definition of connections is done by so-called circuits. One circuit is used per connection.

In the config.txt example two circuits are defined:

- Circuit 1: Dialout over Internet-by-call provider Microsoft Network, Sync-PPP
- Circuit 2: Dialin/Dialout to an ISDN-router (maybe another fli4l) over Raw-IP, i.e. as a connection to a company net somewhere.

If fli4l is simply used as an internet gateway only one circuit is needed. Exception: fli4l's least-cost features should be used. In this case define different circuits for all allowed timespans, see below.

ISDN_CIRC_N Sets the number of used ISDN circuits. If fli4l is used only to monitor incoming ISDN calls set:

```
ISDN_CIRC_N='0'
```

If fli4l is simply used as an internet gateway one circuit is enough. Exception: LC-routing, see below.

ISDN_CIRC_x_NAME Set a name for the circuit - maximum length is 15 characters. The imon client `imonc.exe` will show this instead of the telephone number dialed. Possible characters are 'A' to 'Z' (Capitals are possible), number '0' to '9' and hyphens '-'. Example:

```
ISDN_CIRC_x_NAME='msn'
```

This name can be used in the packet filter or with OpenVPN. If for example the packet filter should control an ISDN circuit a 'circuit_' has to prefix the circuit name. If an ISDN circuit is called 'willi' the packet filter has to be set like this:

```
PF_INPUT_3='if:circuit_willi:any prot:udp 192.168.200.226 192.168.200.254:53 ACCEPT'
```

ISDN_CIRC_x_USEPEERDNS This determines wether the name servers transferred by the internet provider during dial-in should be filled in the configuration file of the local name server for the duration of the connection. This only makes sense for circuits used for connecting to an internet provider. Nearly all providers support this name server tansfer.

After name server IP addresses have been transferred name servers entries from base.txt's *DNS_FORWARDERS* are removed from the configuration file of the local name server and the transferred ones are filled in as forwarders. After this the name server is forced to relaod is configuration. The name server cache will be preserved and names already resolved are kept.

This option has the advantage to work with the nearest possible name servers at any time, as far as the provider transmits correct IP addresses - name resolution is faster then.

In case of failing DNS servers at the provider side transmitted DNS server addresses usually are corrected rapidly by the provider.

After all it is absolutely necessary for the first dial-in to provide a valid name server in base.txt's *DNS_FORWARDERS*. Otherwise the first request can not be resolved correctly. In addition the initial configuration of the name server will be restored on hangup.

Default setting: `ISDN_CIRC_x_USEPEERDNS`='yes'

ISDN_CIRC_x_TYPE ISDN_CIRC_x_TYPE specifies the type of connection x. Possible values are:

'raw' RAW-IP
'ppp' Sync-PPP

In most cases PPP is used, Raw-IP is a little more efficient because of the missing PPP overhead. Authentification is not possible with Raw-IP but with variable ISDN_CIRC_x_DIALIN (see below) limitiations to explicit ISDN numbers ("Clip") can be accomplished. If ISDN_CIRC_x_TYPE is set to'raw' /etc/ppp a raw up/down script will be executed in analog to the PPP up/down scripts.

ISDN_CIRC_x_BUNDLING For ISDN channel bundeling the MPPP protocol according to RFC 1717 is used. This inherits the following mostly irrelevant limitiations:

- Only possible with PPP connections, not with raw circuits
- channel bundeling according to newer RFC 1990 (MLP) is not possible

The second channel either can be added manually using imonc client or automatically by bandwidth adaption, see description for ISDN_CIRC_x_BANDWIDTH.

Default setting: ISDN_CIRC_x_BUNDLING='no'

Caution: using channel bundeling together with compression can trigger some problems, see description for ISDN_CIRC_x_FRAMECOMP.

ISDN_CIRC_x_BANDWIDTH If ISDN channel bundeling is activated by ISDN_CIRC_x_BUNDLING='yes' an automatical addition of the second ISDN channel can be configured here. Two parameters have to be set:

1. threshold level in bytes/second (S)
2. time interval in seconds (Z)

If treshold level S is exceeded for Z seconds imond will add a second channel automatically. If treshold level S is underrun for Z seconds imond will deactivate the second channel again. Automatic bandwidth adaption may be deactivated with ISDN_CIRC_1_BANDWIDTH=''. After that channel bundeling can only be accomplished manually by the imonc client.

Examples:

- ISDN_CIRC_1_BANDWIDTH='6144 30'

  If the transfer rate exceeds 6 kibibyte/second for 30 seconds the second channel will be added.

- ISDN_CIRC_1_BANDWIDTH='0 0'

  The second ISDN channel will be added immedeately (not later than 10 seconds after connection establishment and stays active until the connection terminates completely.

- ISDN_CIRC_1_BANDWIDTH=''

  The second ISDN channel only can be added manually, furthermore ISDN_CIRC_1_BUNDLING='yes' has to be set.

- `ISDN_CIRC_1_BANDWIDTH='10000 30'`

  This is intended to add a second channel after 30 seconds if 10000 B/s were reached during that timespan. This won't work because ISDN has a maximum tranfer rate of 8 kB/s.

If `ISDN_CIRC_x_BUNDLING='no'` is set the value in variable `ISDN_CIRC_x_BANDWIDTH` is irrelevant.

Default setting: `ISDN_CIRC_x_BANDWIDTH=''`

ISDN_CIRC_x_LOCAL This variable holds the local IP address on the ISDN side.

This value should be **empty** if using dynamical address assignment. The IP address will be negotiated during connection establishment. In most cases internet providers hand out dynamic addresses. If a fixed IP address is used specify it here. This variable is optional and has to be added to the config file.

ISDN_CIRC_x_REMOTE This variable holds the remote IP address and netmask on the ISDN side. Classes Inter-Domain routing (CIDR) notation has to be used. Details for CIDR (Page 38) can be found in the base documentation for IP_NET_x.

With dynamic address negotiation this should **empty**. The IP address will be negotiated on connection establishment. In most cases internet providers hand out dynamic addresses. If a fixed IP address is used specify it here. This variable is optional and has to be added to the config file.

The netmask provided will be used for interface configuration after dial-in. A route to the dial-in host itself will be generated as well. As you most probably won't need this route it is best to generate a direct route to the dial-in host by setting the netmask to /32. For details see Chapter: Technical Details For Dialin (Page 364).

ISDN_CIRC_x_MTU ISDN_CIRC_x_MRU With this optional variable the so-called **MTU** (maximum transmission unit) and **MRU** (maximum receive unit) can be set. Optional means that the variable has to be added manually to the configuration file by the user! Usually MTU is 1500 and MRU 1524. This settings should only be changed in rare special cases!

ISDN_CIRC_x_CLAMP_MSS Set this to 'yes' when using synchronous ppp (`ISDN_CIRC_x_TYPE='ppp'`) and one of the following symptoms occurs:

- Webbrowser connects to the webserver without error messages but no pages are displayed and nothing happens,
- sending of small E-mails is working but bigger ones trigger problems or
- ssh works but scp hangs after initial connection.

Default setting: `ISDN_CIRC_x_CLAMP_MSS='no'`

ISDN_CIRC_x_HEADERCOMP `ISDN_CIRC_x_HEADERCOMP='yes'` activates Van-Jacobson compression or header compression. Not all providers are supporting this. If activated compression leads to problems while dialing in set this to 'no'.

Default setting: `ISDN_CIRC_x_HEADERCOMP='yes'`

ISDN_CIRC_x_FRAMECOMP (EXPERIMENTAL) This parameter is only used if `OPT_ISDN_COMP` is set to 'yes'. It handles frame compression.

The following values are possible:

'no'	no frame compression
'default'	LZS according to RFC1974(std) and BSDCOMP 12
'all'	Negotiate lzs and bsdcomp
'lzs'	Negotiate lzs only
'lzsstd'	LZS according to RFC1974 Standard Mode ("Sequential Mode")
'lzsext'	LZS according to RFC1974 Extended Mode
'bsdcomp'	Negotiate bsdcomp only
'lzsstd-mh'	LZS Multihistory according to RFC1974 Standard Mode ("Sequential Mode")

You have to find out by yourself which value is supported by the provider. T-Online supports only 'lzsext' as far as I know. With most other providers 'default' should work.

Attention: using channel bundeling together with 'lzsext' can lead to problems specific to the dial-in server and provider. As providers use different types of dial-in servers there can be differences between dial-in servers of the same provider.

'lzsstd-mh' is meant for router-to-router usage (r2r). It is not used by providers but using it between two fli4l routers leads to significant improvements while transferring more files in parallel. Header compression is needed here and therefore will be activated automatically.

ISDN_CIRC_x_REMOTENAME This variable normally is only relevant when configuring fli4l as a dial-in router. Set the name of a remote hosts if you want but this is not needed.

Default setting: `ISDN_CIRC_x_REMOTENAME`="

ISDN_CIRC_x_PASS Enter provider data here. In the example data for Microsoft Network is used.

`ISDN_CIRC_x_USER` holds the user-id, `ISDN_CIRC_x_PASS` the password.

Note for T-Online:

Username AAAAAAAAAAAATTTTTT#MMMM is composed from a 12 digit 'Anschlußkennung' plus T-Online-Number and 'Mitbenutzernummer'. Put a '#' after the T-Online-Nummer if it is shorter than 12 characters.

In rare cases another '#' character has to be inserted between 'Anschlußkennung' and T-Online-Number.

For T-Online-Numbers with 12 characters no additional '#' is needed.

Example: `ISDN_CIRC_1_USER`='123456#123'

For Raw-IP circuits this variable has no meaning.

ISDN_CIRC_x_ROUTE_N Number of routes of this ISDN circuit. If the circuit defines a default route you must set this to '1'.

ISDN_CIRC_x_ROUTE_X Route(s) for this circuit. For Internet access the first entry should be '0.0.0.0/0' (default route). Format is always 'network/netmaskbits'. A host route for example would look like this: '192.168.199.1/32'. If dialing in to company or university routers name only the net you want to reach there. Examples:

```
ISDN_CIRC_%_ROUTE_N='2'
ISDN_CIRC_%_ROUTE_1='192.168.8.0/24'
ISDN_CIRC_%_ROUTE_2='192.168.9.0/24'
```

All nets must have an explicit entry hence for each route a new ISDN_CIRC_x_ROUTE_y=" line has to be provided.

For using fli4l's LC routing features a default route can be assigned to *several* circuits. Which circuit is used is driven by `ISDN_CIRC_x_TIMES`, see below.

ISDN_CIRC_x_DIALOUT `ISDN_CIRC_x_DIALOUT` specifies the telephone number to be dialed. It is possible to put in several numbers (if one is busy the next is chosen) - numbers have to be separated by blanks. A maximum of five numbers can be used.

ISDN_CIRC_x_DIALIN If the circuit (also) serves for dial-in `ISDN_CIRC_x_DIALIN` keeps the phone number of the callee - with a region prefix but *without* a leading 0. Ports behind telephone systems may have to specify one or even two zeros.

If more users should be able to dial in over a circuit more numbers may be added separated by blanks. It is advised to assign a separate circuit for each caller although. Otherwise two callers trying to dial in at the same time (which is absolutetely feasible with 2 ISDN channels) could collide on behalf of IP addresses assigned.

If callers don't transfer a number during calling '0' could be used. Caution: everyone not transferring a number is allowed to call in then!

If number-independent dial-in should be realized set '*'.

In both cases a separate authentification (see `ISDN_CIRC_x_AUTH`) is unavoidable.

ISDN_CIRC_x_CALLBACK Settings for callback, possible values:

'in'	fli4l is called and calls back
'out'	fli4l calls, hangs up and waits for callback
'off'	no callback
'cbcp'	callback control protocol
'cbcp0'	callback control protocol 0
'cbcp3'	callback control protocol 3
'cbcp6'	callback control protocol 6

CallBack control protocol (aka 'Microsoft CallBack') cbcp6 is the protocol mostly used.

Default setting: 'off'

ISDN_CIRC_x_CBNUMBER Set a callback number for protocol cbcp, cbcp3 and cbcp6 here (mandatory for cbcp3).

ISDN_CIRC_x_CBDELAY This variable sets a delay in seconds to be waited until triggering callback. The meaning differs depending on the direction of callback:

- `ISDN_CIRC_x_CALLBACK`='in':

  If fli4l is called and should call back `ISDN_CIRC_x_CBDELAY` specifies the waiting time until calling back. Use `ISDN_CIRC_x_CBDELAY`='3' as a rule of thumb. A lower value may work and speed up connection establishment then depending on whom to call back.

- ISDN_CIRC_x_CALLBACK='out':

  In this case ISDN_CIRC_x_CBDELAY is the ringing timespan for the other party until fli4l waits for callback. ISDN_CIRC_x_CBDELAY='3' is a good rule of thumb here either. On long distance calls it takes up to 3 seconds untils the other router is even recognizing the call. If in doubt simply try.

If setting ISDN_CIRC_x_CALLBACK='off', ISDN_CIRC_x_CBDELAY is ignored. This variable is ignored with CallBack Control Protocol as well.

ISDN_CIRC_x_EAZ In the example the MSN (called EAZ here) is set to 81330. Set your own MSN *without* area code here.

Depending on your telephony system only the extension station number could be necessary. Setting an additional '0' may also help sometimes.

This variable may be empty which can help with some telephony systems as well.

ISDN_CIRC_x_SLAVE_EAZ If fli4l is connected on the internal S0-Bus of a telephony system and you want to use channel bundeling you may have to specify a second extension station number for the slave channel.

Normally this variable can stay empty.

ISDN_CIRC_x_DEBUG If ipppd should display additional debug informations set ISDN_CIRC_x_DEBUG to 'yes'. Ipppd will log these informations to syslog then.

IMPORTANT: To use syslogd for logging OPT_SYSLOGD has to be set to 'yes'.
(See OPT_SYSLOGD - Program For Protocolling System Messsages (Page 71)) Some messages are logged to klogd so OPT_KLOGD (See OPT_KLOGD - Kernel-Message-Logger (Page 73)) should be set to 'yes' as well for debugging ISDN.

For Raw-IP-Circuits ISDN_CIRC_x_DEBUG has no meaning.

ISDN_CIRC_x_AUTH If the circuit is also used for dial-in and an authentification over PAP or CHAP should be used by the calling party set ISDN_CIRC_x_AUTH to 'pap' or 'chap' - but *only* then. In all other cases this variable has to be empty!

Cause: An Internet provider will never authentificate with you - but there may be exceptions to this rule.

Use the entries ISDN_CIRC_x_USER and ISDN_CIRC_x_PASS for username and password.

For Raw-IP-Circuits this variable has no meaning.

ISDN_CIRC_x_HUP_TIMEOUT ISDN_CIRC_x_HUP_TIMEOUT sets the time after that fli4l disconnects from the provider if no traffic is detected over the connection. In the example the connection will be hung up after 40 seconds idle time to save money. On new accesses fli4l connects again in a short timespan. This makes sense especially with providers that have seconds charge intervals!

At least while testing you should keep an eye on fli4l's automatic dial-in and hangup using either console or imon-client. In case of faulty configuration the ISDN connection could become an unwanted permanent line.

Setting this to '0' means that fli4l doesn't use idle time and won't hang up by itself anymore. Use with care!

ISDN_CIRC_x_CHARGEINT Set charge interval in seconds which will be used for calculating online costs.

Most providers charge by minute intervals. In this case use the value '60'. For providers that charge in seconds set `ISDN_CIRC_x_CHARGEINT` to '1'.

Addition for `ISDN_CIRC_x_CHARGEINT` >= 60 seconds:

If no traffic was detected for `ISDN_CIRC_x_HUP_TIMEOUT` seconds the connection will be terminated approx. 2 seconds before reaching the chargint timespan. Charged time is used nearly complete this way. A really neat feature of fli4l.

With charging intervals of under a minute this does not make sense so this feature is only used for charging intervals of more than 60 seconds.

ISDN_CIRC_x_TIMES Specify here when and at what charge the circuit should be active. This makes it possible to use different circuits as default routes at different daytimes (least-cost-routing).The imond daemon controls route-allocation.

Composition of the variable:

```
ISDN_CIRC_x_TIMES='times-1-info [times-2-info] ...'
```

Each times-?-info field consists of 4 subfields separated by colons (':').

1. Field: W1-W2

   Weekday-timespan, for example Mo-Fr or Sa-Su. English and german notation are possible. If a single weekday should be specified write W1-W1, for example Su-Su.

2. Field: hh-hh

   Daytime-timespan, for example 09-18 or also 18-09. 18-09 is equal to 18-24 plus 00-09. 00-24 means the whole day.

3. Field: Charge

   Costs per minute in currency units, for example 0.032 for 3.2 Cent per minute. The real costs are calculated in consideration of charging intervals and displayed in imon-client then.

4. Field: LC-default-route

   May be Y or N. Meaning:

   - Y: The timespan specified will be used as default route for LC-routing. Important: in this case `ISDN_CIRC_x_ROUTE`='0.0.0.0/0' must be set in addition!
   - N: The timespan specified only serves for calculating online costs but won't be used for LC-Routing.

Example:

```
ISDN_CIRC_1_TIMES='Mo-Fr:09-18:0.049:N Mo-Fr:18-09:0.044:Y Sa-Su:00-24:0.044:Y'
ISDN_CIRC_2_TIMES='Mo-Fr:09-18:0.019:Y Mo-Fr:18-09:0.044:N Sa-Su:00-24:0.044:N'
```

Read as follows: Circuit 1 should be used on labour days evenings/nights and on the complete weekends, Circuit 2 is used on labour days from 9 AM to 6 PM.

Important: *timespans specified in* `ISDN_CIRC_x_TIMES` *have to cover the whole week. Without that no valid configuration can be generated.*

Important: *If timespans of all LC-default-route circuits ("Y") don't cover the complete week no default route exists during the missing times. Therefore no internet connections are possible!*

Example:

```
ISDN_CIRC_1_TIMES='Sa-Su:00-24:0.044:Y Mo-Fr:09-18:0.049:N Mo-Fr:18-09:0.044:N'
ISDN_CIRC_2_TIMES='Sa-Su:00-24:0.044:N Mo-Fr:09-18:0.019:Y Mo-Fr:18-09:0.044:N'
```

Here for labour days from 18-09 "N" is set. At this times no route to the internet exists - surfing forbidden!

Another simple example:

```
ISDN_CIRC_1_TIMES='Mo-Su:00-24:0.0:Y'
```

for those using a flatrate.

A last note concerning LC-Routing:

Bank holidays are treated as sundays.

4.13.5. OPT_TELMOND - telmond-Configuration

`OPT_TELMOND` determines wheter the telmond server is activated or not. It listens to incoming telephone calls and signals on TCP port 5001 the caller id transmitted and called. This information can be queried and displayed by i.e. windows- or Unix/Linux imon-client (see chapter "Client-/Server-interface imond").

An installed ISDN card is mandatory as well as as valid configuration of `OPT_ISDN`.

Testing the correct function of telmond is possible with:

```
telnet fli4l 5001
```

This should display the last call and immedeately close the telnet connection.

Port 5001 is only reachable from LAN. Access from outside is blocked by the firewall per default. Further access limitations are configurable via telmond variables, see below.

Default setting: `START_TELMOND`='yes'

TELMOND_PORT TCP/IP-Port on which telmond listens for connections. The default setting '5001' should only be changed in rare exceptions.

TELMOND_LOG `TELMOND_LOG`='yes' activates saving of all incoming calls in a file called /var/log/telmond.log. The content of this file can be queried with imond-Client imonc under Unix/Linux and Windows.

Different paths or logfiles splitted by clients may be configured below.

Default setting: `TELMOND_LOG`='no'

TELMOND_LOGDIR If protocolling is active `TELMOND_LOGDIR` can set a different path instead of /var/log, for example '/boot'. The file telmond.log will be saved on the boot media (which has to be mounted Read/Write "rw") then. If 'auto' is set the logfile is created in

/boot/persistent/isdn or at another path specified by FLI4L_UUID. If /boot is not mounted Read/Write the file is created in /var/run.

TELMOND_MSN_N If certain calls should only be visible on some client PC's imonc a filter can be set to achieve that MSNs are only protocolled for those PCs.

If this is necessary (for example with flat sharing) the variable TELMOND_MSN_N holds the number of MSN filters.

Default setting: TELMOND_MSN_N='0'

TELMOND_MSN_x For each MSN filter a list of IP addresses has to be set which should be able to view the call informations.

The variable TELMOND_MSN_N determines the number of those configurations, see above.

Composition of the variable:

```
TELMOND_MSN_x='MSN IP-ADDR-1 IP-ADDR-2 ...'
```

A simple example:

```
TELMOND_MSN_1='123456789 192.168.6.2'
```

If a call for a certain MSN should be displayed on more computers their IP addresses have to be specified one after the other.

```
TELMOND_MSN_1='123456789 192.168.6.2 192.168.6.3'
```

TELMOND_CMD_N If a telephone call (Voice) is coming in for a MSN some commands can be executed on the fli4l router optionally. TELMOND_CMD_N holds the number of commands to be executed.

TELMOND_CMD_x TELMOND_CMD_1 bis TELMOND_CMD_n holds commands to be executed for an incoming phone call.

Variable TELMOND_CMD_N specifies the quantity of commands, see above.

Composition of the variable:

```
MSN CALLER-NUMBER  COMMAND ...
```

Set the MSN without area prefix. CALLER-NUMBER takes the complete caller id with area prefix. If CALLER-NUMBER is set to an asterisk (*) telmond won't pay attention to the caller id.

An example:

```
TELMOND_CMD_1='1234567 0987654321 sleep 5; imonc dial'
TELMOND_CMD_2='1234568 * switch-on-coffee-machine'
```

In the first case the command sequence "sleep 5; imonc dial" is executed if caller with id 0987654321 calls MSN 1234567. Two commands are executed. At first fli4l will wait for 5 seconds for the ISDN channel to become available. After that the fli4l client imonc is started with the argument "dial". imonc passes this command to the telmond server which will establish a network connection on the default route circuit. Which commands the imonc client can pass to the imond server is described in chapter "Client-/Server interface imond". `OPT_IMONC` from the package "tools" has to be installed to get this working.

The second command "switch-on-coffee-machine" will be executed if a call on MSN 1234568 comes in independent on caller id. The command "switch-on-coffee-machine" does not exist for fli4l (at the moment)!

On execution of command the following placeholders may be used:

%d	date	Date
%t	time	Time
%p	phone	Caller ID
%m	msn	Own MSN
%%	percent	the percent sign

This data can be used by the programs called, for example for sending E-Mail.

TELMOND_CAPI_CTRL_N If using a CAPI capable ISDN adapter or a remote CAPI (type 160 or 161) it may be necessary to configure the CAPI controller on which telmond is listening for calls more explicitly. For example the Fritz!Box offers access to up to five different controllers which may not even differ (see informations at http://www.wehavemorefun.de/fritzbox/CAPI-over-TCP#Virtuelle_Controller). To limit the number of controllers to be used you may set the quantity. In the following array-variables `TELMOND_CAPI_CTRL_%` it may be set which controllers are to be used.

If you don't use this variable telmond will listen on *all* available CAPI controllers.

TELMOND_CAPI_CTRL_x If `TELMOND_CAPI_CTRL_N` is unequal to zero the indices for the CAPI controllers have to be specified on which telmond should monitor incoming calls.

Example for the remote CAPI of a Fritz!Box with "real" ISDN connection:

```
TELMOND_CAPI_CTRL_N='2'
TELMOND_CAPI_CTRL_1='1' # listen to incoming ISDN calls
TELMOND_CAPI_CTRL_2='3' # listen to calls on the internal S0-Bus
```

Example for the remote CAPI of a Fritz!Box with analog connection and SIP-Forwarding:

```
TELMOND_CAPI_CTRL_N='2'
TELMOND_CAPI_CTRL_1='4' # listen to incoming analog calls
TELMOND_CAPI_CTRL_2='5' # listen to incoming SIP-calls
```

4.13.6. OPT_RCAPID - Remote CAPI Daemon

This OPT configures the program rcapid on the fli4l router which offers access to the ISDN CAPI interface of the routers via network. Appropritate tools can access the ISDN card of

the routers via network as if it was installed locally. This is similar to the package "mtgcapri". The difference is that only Windows systems can use "mtgcapri" as a client while the network interface of rcapid is only supporting linux systems at the time of writing. So both packages are ideal complements in mixed Windows- and Linux environments.

Konfiguration des Routers

OPT_RCAPID This variable activates offering of the router's ISDN-CAPI for remote clients. Possible values are "yes" and "no". If set to "yes" the internet daemon inetd will start the rcapid daemon on incoming queries on rcapid port 6000 (port my be changed using variable `RCAPID_PORT`).

Example: `OPT_RCAPID='yes'`

RCAPID_PORT This variable holds the TCP port to be used by the rcapid daemon.

Default setting: `RCAPID_PORT='6000'`

Configuration Of Linux Clients

To use the remote CAPI interface on a Linux PC the modular libcapi20 library must be used. Actual Linux distributions install such a CAPI library (i.e. Debian Wheezy). If not the sources may be downloaded from http://ftp.de.debian.org/debian/pool/main/i/isdnutils/isdnutils_3.25+dfsg1.orig.tar.bz2. After unpacking and changing to the directory "capi20" the CAPI library may be compiled and installed with "./configure", "make" and "sudo make install" as usual. If the library is installed the configuration file `/etc/capi20.conf` has to be adapted to specifiy the client on which rcapid is running. If the router for example is reached by the name of "fli4l" the conf file will looks as follows:

```
REMOTE fli4l 6000
```

That's all! If the program "capiinfo" is installed on the linux client (part of capi4k-utils-package of many distributions) you can test the remote CAPI interface:

```
kristov@peacock ~ $ capiinfo
Number of Controllers : 1
Controller 1:
Manufacturer: AVM Berlin
CAPI Version: 1073741824.1229996355
Manufacturer Version: 2.2-00  (808333856.1377840928)
Serial Number: 0004711
BChannels: 2
[...]
```

Under "Number of Controllers" the quantity of ISDN cards is displayed which are usable on the client. If this reads "0" the connection to the rcapid program is working but the ISDN card is not recognized on the router. If the connection to the rcapid program is not working at all (maybe `OPT_RCAPID` is set to "no") an error message "capi not installed - Connection refused (111)" will be displayed. In this case check your configuration once more.

4.14. OpenVPN - VPN Support

As of version 2.1.5 package OpenVPN is part of fli4l.

Important: *For using OpenVPN over the Internet a flatrate or billing based on data volume is a must have! If the fli4l router is powered on the connection will never be hung up because a small amount of data is permanently transferred by OpenVPN. Using a VPN Tunnel over the Internet thus can cause high online costs. The same is applying for an ISDN connection being used for OpenVPN.*

Besides OpenVPN another VPN package exists: OPT_PoPToP (see opt-database http://www.fli4l.de/download/zusatzpakete/).

Deciding which VPN solution to use is driven by security and function concerns. No advices on security of the different packages are given by the team. In unsure, see

Linux-Magazine January 2004

http://diswww.mit.edu/bloom-picayune/crypto/14238

http://sites.inka.de/bigred/archive/cipe-l/2003-09/msg00263.html

Concerning functionality a clear advice can be given to use OpenVPN which outperforms both CIPE and poptop here. OpenVPN supports tunnel mode, bridge mode, data compression and is more solid than CIPE on a fli4l router. OpenVPN has a Windows version to be used as of Windows 2000. Only disadvantages against CIPE are the sheer size in opt archive and missing OpenVPN support for fli4l version 2.0.x.

4.14.1. OpenVPN - Introductive Example

To introduce you to OpenVPN's configuration at first a small example. Two networks that both use a fli4l router shall be connected over the Internet. OpenVPN establishes an encrypted tunnel on both fli4l routers to let computers from both nets communicate with each other. The configuration variables shown in picture 4.1 are used for this purpose.

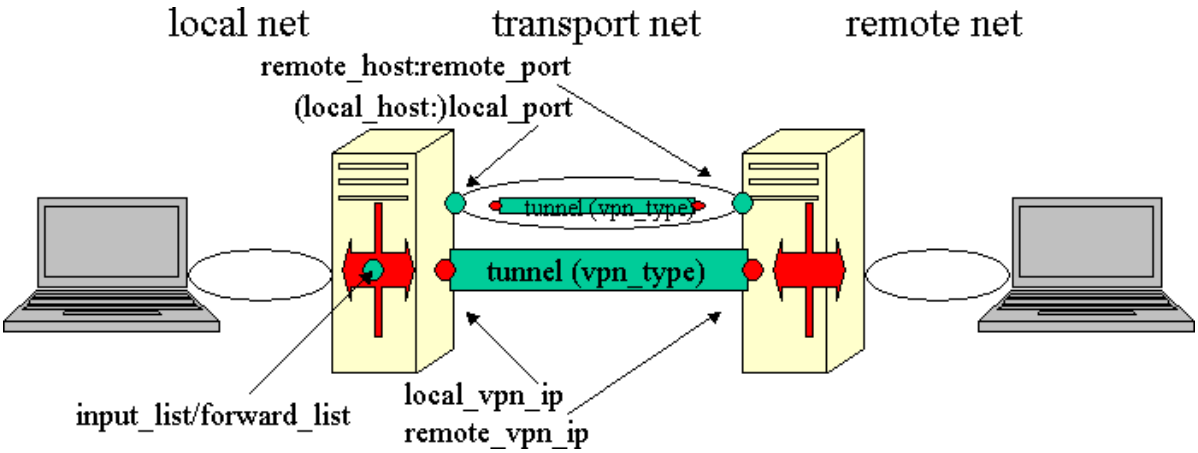


Figure 4.1.: VPN configuration example — tunnel between two routers

local net, remote net represent the two nets to be connected by the tunnel. They have to be in different TCP/IP ranges and should have different non interfering net masks. The settings from IP_NET_x (Page 38) in both routers base.txt configuration files hence have

to be different. Thus it is not possible to connect two nets over a tunnel that both use IP range 192.168.6.0/24.

transport net The transport network consists of two elements:

- the connection between two OpenVPN daemons, described by *remote_host:remote_port* and *(local_host:)local_port*. This is an equivalent to the OpenVPN settings in `OPENVPN_x_REMOTE_HOST`, `OPENVPN_x_REMOTE_PORT`, `OPENVPN_x_LOCAL_HOST` and `OPENVPN_x_LOCAL_PORT`.
- and a tunnel over which the connection between the two OpenVPN-Daemons is established, described by *local_vpn_ip/remote_vpn_ip*. This again is an equivalent to the OpenVPN settings in `OPENVPN_x_LOCAL_VPN_IP` and `OPENVPN_x_REMOTE_VPN_IP`. Both VPN IP addresses have be set to values of non existent nets on both routers.

input_list, forward_list Packets that should be sent over this tunnel have to pass the packet filter at first. It will only allow ICMP messages (i.e. ping) as the default which can be used to test the tunnel. Everythihng else has to be allowed explicitly. In the simplest case this is done by

```
OPENVPN_x_PF_INPUT_POLICY='ACCEPT'
OPENVPN_x_PF_FORWARD_POLICY='ACCEPT'
```

Plese note that „accepting“ a complete VPN connection is very critical in terms of security. Better use the tmpl: syntax of the packet filter to only allow those services needed.

No more settings are required for a simple VPN tunnel. All other configuration handles extended functions or special use cases. You should use those after establishing a working tunnel with this minimal configuration.

4.14.2. OpenVPN - Configuration

Because of the complexity of OpenVPN we start by explaining settings required for any VPN connection. Don't try extended configurations for OpenVPN before establishing a connection with minimal settings.

OPT_OPENVPN Default: `OPT_OPENVPN='no'`

`'yes'` activates package OpenVPN. `'no'` deactivates package OpenVPN completely.

OPENVPN_N Default: `OPENVPN_N='0'`

How many OpenVPN configurations are active in the configuration file?

OPENVPN_x_REMOTE_HOST Default: `OPENVPN_x_REMOTE_HOST="`

IP address or DNS address of the remote OpenVPN. For a Roadwarrior (Page 179) this line has to be completely omitted. If omitted OpenVPN waits for connection establishment and doesn't try to connect by itself.

OPENVPN_x_REMOTE_HOST_N Default: `OPENVPN_x_REMOTE_HOST_N='0'`

Using dynamic DNS services is not alsways 100% reliable. You may simply use two ore more of those DynDNS services to register your current IP address with all of them at the same time. To enable OpenVPN to go through the whole DynDNS names a list of *additional* DNS names has to be set. By the help of `OPENVPN_x_REMOTE_HOST` OpenVPN will try to contact these addresses in random order. Hence `OPENVPN_x_REMOTE_HOST` has to exist and be configured correctly!

OPENVPN_x_REMOTE_HOST_x Default: `OPENVPN_x_REMOTE_HOST_x=''`

Same description as above applies here `OPENVPN_x_REMOTE_HOST` (Page 161).

OPENVPN_x_REMOTE_PORT Default: `OPENVPN_x_REMOTE_PORT=''`

Each OpenVPN connection does need an unused port address on the fli4l router. It is adviced to use ports above 10000 for those are not commonly used. If configuring a connection for a remote station with dynamically changing IP address that has no DynDNS address omit this entry as well as `OPENVPN_x_REMOTE_HOST`.

OPENVPN_x_LOCAL_HOST Default: `OPENVPN_x_LOCAL_HOST=''`

Specifies to what IP address OpenVPN will bind. For connections over the Internet this entry should be completely omitted. If an address is set here OpenVPN will only listen for incoming traffic on this IP. If you want to secure a WLAN connection you should set the IP address of fli4l's WLAN interface card here.

OPENVPN_x_LOCAL_PORT Default: `OPENVPN_x_LOCAL_PORT=''`

Specifies the port number the local OpenVPN daemon will listen to. For each OpenVPN connection you need a reserved port that only can be used by this connection. Other software on the router is not allowed to use this port. `OPENVPN_x_REMOTE_PORT` and `OPENVPN_x_LOCAL_PORT` of each OpenVPN connection have to match! If setting `OPENVPN_x_REMOTE_PORT='10111'` on one side of the tunnel `OPENVPN_x_LOCAL_PORT='10111'` *has to be set* on the other side as well.

Again: It is very important to match these settings to the according remote OpenVPN station otherwise a connection is not possible between OpenVPN partners.

To enable OpenVPN to listen to incoming connections OpenVPN itself opens the ports in the packet filter set in `OPENVPN_x_LOCAL_PORT`. If this is not your wish then you may change this behavior in `OPENVPN_DEFAULT_OPEN_OVPNPORT` (Page 168). It is *not* necessary to set `OPENVPN_DEFAULT_OPEN_OVPNPORT='yes'` because this is the default behavior!

OpenVPN does not work with ports lower than 1025. If i.e. OpenVPN should work as a tcp-server on port 443 (https port) you have to forward this port via the packet filter to a port above 1024. If i.e. OpenVPN is listening on port 5555 and port 443 should be forwarded there `PF_PREROUTING` has to be set like this:

```
PF_PREROUTING_5='tmpl:https dynamic REDIRECT:5555'
```

OPENVPN_x_SECRET Default: `OPENVPN_x_SECRET=''`

OpenVPN needs a keyfile for encrypting an OpenVPN connection. This keyfile can be generated unter Windows or Linux by OpenVPN itself. Beginners may install OpenVPN's Windows software or use OpenVPN's WebGUI. If you do not want to use OpenVPN under Windows but only generate the needed keyfiles it is enough to install *OpenVPN User-Space Components*, *OpenSSL DDLs*, *OpenSSL Utilities*, *Add OpenVPN to PATH* and *Add Shortcuts to OpenVPN*. With choosing *Generate a static OpenVPN key* from the OpenVPN start menu the keyfiles needed can be generated. At the end the message „Randomly generated 2048 bit key written to `C:/Program files/OpenVPN/config/key.txt`" will appear. The file `key.txt` is the one we need. Copy this file into the directory `<config>/etc/openvpn` and change its name `key.txt` to something more meaningful. Keyfiles can also be generated automatically by the fli4l router if you set `OPENVPN_CREATE_SECRET` to `'yes'` and reboot fli4l. If configuring OpenVPN for the first time enter all data in the config file and either set `OPENVPN_DEFAULT_CREATE_SECRET` (Page 167) to `'yes'` if one keyfile should be used for all connections or if a keyfile for only one connection should be generated set `OPENVPN_x_CREATE_SECRET` to `'yes'`. After boot of the fli4l router one or more keyfiles will be created automatically and saved to `/etc/openvpn` with the name specified. Keyfile(s) can be copied via scp or other medias. After creation of keyfiles change the setting back to 'no' and build a new boot media for fli4l with the configuration and keyfiles you just created. If you forget to change 'yes' to 'no' fli4l will generate new keyfiles with each reboot but no OpenVPN daemon will be started and thus no tunnels can be established. If you set `OPENVPN_x_CREATE_SECRET` to `'webgui'` you can use the web interface to generate keyfiles. Use OpenVPN's WebGUI in detail view for connections and choose 'Keymanagement'. For reference see 4.14.6

Hint: By executing

```
openvpn --genkey --secret <filename>
```

you can generate a keyfile by hand via fli4l's console.

Keyfiles have to be copied to the directory `<config>/etc/openvpn` as seen in the following picture. The file name of the keyfiles without path has to set in `OPENVPN_x_SECRET`. In this way keyfiles will be copied to the opt-archive while creating the boot media.

OPENVPN_x_TYPE Default: `OPENVPN_x_TYPE="`

An OpenVPN connection either can be used as a tunnel or as a bridge. Through an OpenVPN tunnel only IP traffic can be routed. A bridge transfers ethernet frames i.e. not only IP traffic but also IPX or NetBEUI or else. For using OpenVPN to transfer ethernet frames package advanced_networking is needed in addition. Please note that a bridge over a DSL line can be really slow!

4.14.3. OpenVPN - Bridge configuration

For using OpenVPN as a bridge the following entries are valid. Please note that when using a bridge over the Internet broadcast traffic uses already a rather high bandwidth without any real data being transferred.

Remember that the following settings are only valid if for this connection `OPENVPN_x_TYPE` (Page 163) is set to `'bridge'`! A configured bridge from package advanced_networking to which the VPN connection can bind is needed additionally.

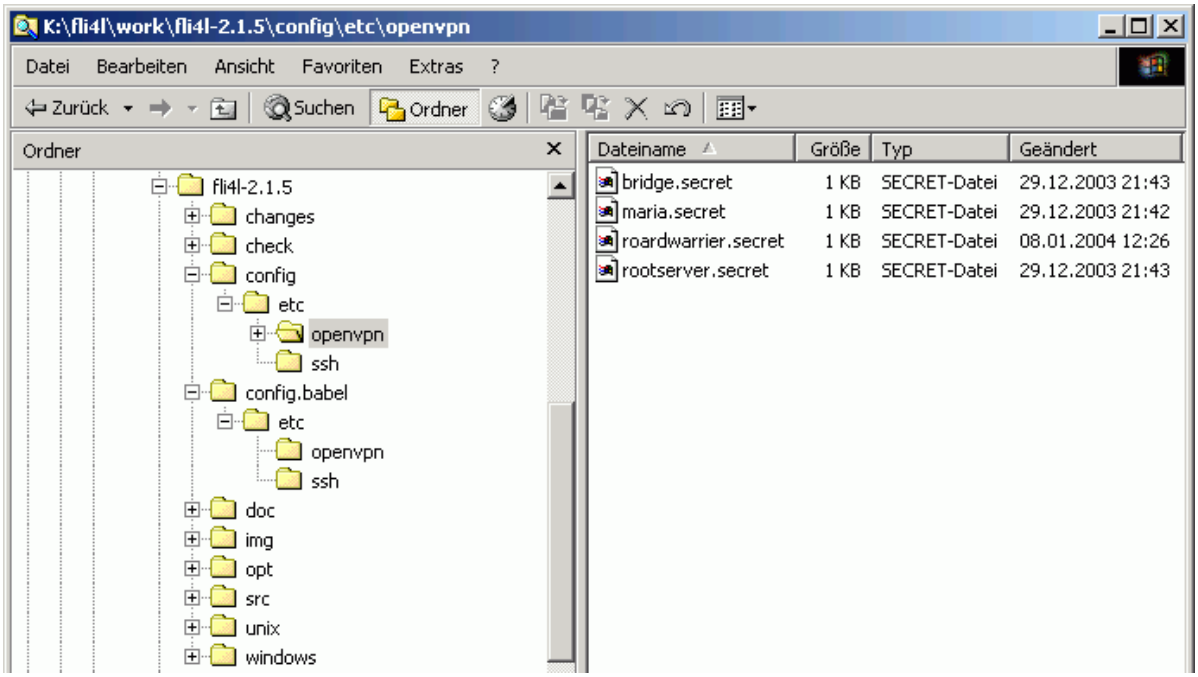


Figure 4.2.: fli4l config directory with OpenVPN *.secret files

OPENVPN_x_BRIDGE Default: `OPENVPN_x_BRIDGE="`

Holds the name of the bridge this OpenVPN connection should bind to. If `BRIDGE_DEV_x_NAME='cuj-br'` is given and the OpenVPN connection should bind to that bridge `'cuj-br'` has to be set in accordance.

OPENVPN_x_BRIDGE_COST Default: `OPENVPN_x_BRIDGE_COST="`

If using spanning tree protocol (STP, see http://de.wikipedia.org/wiki/Spanning_Tree or documentation for package advanced_networking) you can specify the connection costs here.

OPENVPN_x_BRIDGE_PRIORITY Default: `OPENVPN_x_BRIDGE_PRIORITY="`

If using STP (spanning tree protocol, see http://de.wikipedia.org/wiki/Spanning_Tree or documentation for package advanced_networking) you can specify connection priority here.

4.14.4. OpenVPN - Tunnel configuration

OPENVPN_x_REMOTE_VPN_IP Default: `OPENVPN_x_REMOTE_VPN_IP="`

This setting is only valid if `OPENVPN_x_TYPE` (Page 163) is set to `'tunnel'` for this OpenVPN connection!

VPN IP address of the OpenVPN remote station. VPN IP addresses are needed only for routing and can be chosen nearly free. The following restrictions apply:

- IP address may not be used in local network and thus can't be in the subnet of the fli4l router.

- IP address may not be used for any local network device.
- IP address may not belong to any network routed by IP_ROUTE_x.
- IP address may not belong to any network routed by ISDN_CIRC_ROUTE_x.
- IP address may not belong to any network routed by CIPE_ROUTE_x.
- IP address may not belong to any network routed by OPENVPN_ROUTE_x.
- IP address may not belong to any network used or routed by fli4l in any other way.

As you see VPN IP addresses can't be used anywhere else. Before beginning to configure OpenVPN you should look for an unused net in both local and remote address ranges. It should belong to private address ranges (see http://ftp.univie.ac.at/netinfo/rfc/rfc1597.txt).

OPENVPN_x_LOCAL_VPN_IP Default: OPENVPN_x_LOCAL_VPN_IP="

This setting is only valid if OPENVPN_x_TYPE (Page 163) is set to 'tunnel' for this OpenVPN connection.

IP address for the local OpenVPN device tunX. Same restrictions as in OPENVPN_x_REMOTE_VPN_IP (Page 164) apply here.

By the way, it is possible to use the same IP address as in OPENVPN_x_LOCAL_VPN_IP for all local OpenVPN connections. This enables a host host to use the same IP address in all VPNs. Packet filter rules are drastically easier to configure this way.

OPENVPN_x_IPV6 Default: OPENVPN_x_IPV6='no'

This enables native IPv6 support in OpenVPN. Consider this as an experimental feature because of the code being brandnew. Of course OPT_IPV6 has to be activated and configured as well then. For OPENVPN_x_IPV6='no' and/or OPT_IPV6='no' all relevant variables are ignored.

ATTENTION!!! These settings are not checked for overlapping with other parts of the configuration! This applies to OPENVPN_x_LOCAL_VPN_IPV6, OPENVPN_x_REMOTE_VPN_IPV6 and OPENVPN_x_ROUTE_x.

OPENVPN_x_REMOTE_VPN_IPV6 Default: OPENVPN_x_REMOTE_VPN_IPV6="

For IPv6 the same restrictions apply as for OPENVPN_x_REMOTE_VPN_IP (Page 164).

```
OPENVPN_X_REMOTE_IPV6='FD00::1'
```

OPENVPN_x_LOCAL_VPN_IPV6 Default: OPENVPN_x_LOCAL_VPN_IPV6="

For IPv6 applies the same as for OPENVPN_x_LOCAL_VPN_IP (Page 165). If no subnet is set /64 will be used as a default.

```
OPENVPN_X_LOCAL_IPV6='FD00::2/112'
```

OPENVPN_x_ROUTE_N Default: `OPENVPN_x_ROUTE_N=""`

This setting is only valid if `OPENVPN_x_TYPE` (Page 163) is set to `'tunnel'` for this OpenVPN connection.

Routes are being set automatically by OpenVPN when starting up. Up to 50 nets can be routed over a single OpenVPN connection. For every net to be routed a valid `OPENVPN_x_ROUTE_x` entry must be created.

Please note that the packet filter rules necessary have to be set manually in `OPENVPN_PF_FORWARD_x` `OPENVPN_PF_INPUT_x` res. `OPENVPN_PF6_FORWARD_x` `OPENVPN_PF6_INPUT_x`. OpenVPN only allows ICMP over a VPN connection and denies all other data traffic. Details can be found at `OPENVPN_x_PF_INPUT_N` (Page 173) and `OPENVPN_x_PF_FORWARD_N` (Page 174) res. at `OPENVPN_x_PF6_INPUT_N` (Page 174) and `OPENVPN_x_PF6_FORWARD_N` (Page 174).

OPENVPN_x_ROUTE_x Default: `OPENVPN_x_ROUTE_x=""`

Specify the nets to be reached over the OpenVPN remote station here. If on the remote side i.e. the nets 192.168.33.0/24 and 172.18.0.0/16 can be reached and should be accessed through the OpenVPN tunnel both of them have to be entered under `OPENVPN_x_ROUTE_x`. Host routes (`/32`) may be set here as well.

If the default route should be reached through an OpenVPN tunnel specifiy 0.0.0.0/0 res. ::/0 for IPv6 and an optional flag as routes here. For IPv6 routes OPT_IPv6 has to be activated, local and remote IPv6 addresses for the tunnel have to be set and OPENVPN_x_IPV6 must be 'yes'. OpenVPN has several alternative ways to set a default route which can be chosen by a flag. Each method has its own advantages and disadvantages. At the moment the following flags are supported:

local The *local* flag should be chosen if the OpenVPN remote station is located in a subnet that can be reached directly by the fli4l router. This may be the case for example for an OPENVPN default route over WLAN.

def1 With this flag two new routes 0.0.0.0/1 and 128.0.0.0/1 will be defined in addition to a host route to the OpenVPN remote station. This routes act as default routes for the complete (encrypted) traffic to the OpenVPN remote station (which can be reached over the host route).

If omitting the optional flag OpenVPN will choose the method of setting default routes. Methods will be picked by the OpenVPN version. At the moment *local* is the default advised.

```
OPENVPN_1_ROUTE_N='3'
OPENVPN_1_ROUTE_1='192.168.33.0/24'
OPENVPN_1_ROUTE_2='172.18.0.0/16'
OPENVPN_1_ROUTE_3='2001:db8:/32'
```

OpenVPN - Delegation Of DNS and Reverse-DNS

OPENVPN_x_DOMAIN Default: `OPENVPN_x_DOMAIN=""`

This parameter sets the remote domain. The variable can hold multiple domains which have to be separated by spaces then. If only this parameter is set (without mentioning of

an additional DNS server) it will be assumed that a DNS server is listening on the IP of the other end of the tunnel (see OPENVPN_x_REMOTE_VPN_IP (Page 164)). On the remote router incoming DNS queries have to allowed in this case. (i.e. via OPENVPN_x_INPUT_y='tmpl:dns ACCEPT')

OPENVPN_x_ROUTE_x_DOMAIN Default: OPENVPN_x_ROUTE_x_DOMAIN=""

Different subnets can have different domains assigned. Per OPENVPN_x_ROUTE_y one according domain can be configured. If a OPENVPN_x_ROUTE_y_DNSIP exists for the domain, it will be used, else the one set at OPENVPN_x_DNSIP. The effect is the same as with OPENVPN_x_DOMAIN but this method allows better documentation.

OPENVPN_x_DNSIP Default: OPENVPN_x_DNSIP=""

If the tunnel end point is not the appropriate DNS server set the IP of the appropriate one here. If this is empty the one at OPENVPN_x_REMOTE_VPN_IP (Page 164) will be used.

OPENVPN_x_ROUTE_x_DNSIP Default: OPENVPN_x_ROUTE_x_DNSIP=""

Multiple subnets routed can also have different DNS servers - define one per OPENVPN_x_ROUTE_x (Page 166) here.

4.14.5. Expert Settings

Settings described in this chapter are all optional and should only be changed if the OpenVPN connection is working but should be optimized (for example by the use of another encryption algorithm).

All settings in OPENVPN_DEFAULT_ are otional. This means they don't have to be written in the config file. If an entry is missing in openvpn.txt the OpenVPN start script will use the default value described here. If you don't want to change this defaults do not write them to the openvpn.txt config file!

General Settings

OPENVPN_DEFAULT_CIPHER Default: OPENVPN_DEFAULT_CIPHER='BF-CBC'

One of the available encryption methods. Method 'BF-CBC' is used as a default by all OpenVPN versions (also non-fli4l specific versions).

OPENVPN_DEFAULT_COMPRESS Default: OPENVPN_DEFAULT_COMPRESS='yes'

OpenVPN uses adaptive LZO data compression to enlarge the bandwidth of a connection. Adaptive means OpenVPN recognizes by itself when i.e. already compressed zip files are sent over an OpenVPN connection. In such case data compression will be switched off until data is sent that will benefit from data compression. There is nearly no cause for deactivating data compression because this enlarges bandwidth at nearly no cost. Only disadvantage of data compression is a small increase of latency by some milliseconds. For online games via VPN which need a "good" ping, i.e. low latency it may be wise to deactivate data compression.

OPENVPN_DEFAULT_CREATE_SECRET Default: OPENVPN_DEFAULT_CREATE_SECRET='no'

This setting will cause OpenVPN to automatically generate keyfiles on boot of the fli4l router. An OpenVPN connection won't be started then. For details see OPENVPN_x_SECRET (Page 162).

OPENVPN_DEFAULT_DIGEST Default: `OPENVPN_DEFAULT_DIGEST='SHA1'`

Enter available checksums her. OpenVPN uses `'SHA1'` as default.

OPENVPN_DEFAULT_FLOAT Default: `OPENVPN_DEFAULT_FLOAT='yes'`

OpenVPN remote stations that use DynDNS addresses can change their IP address at any time. To make OpenVPN accept this changed IP address set `OPENVPN_DEFAULT_FLOAT` to `'yes'`. If `'no'` is set changing of an IP address is not allowed. This only makes sense with WLAN connections or connections to remote stations with static IP addresses (i.e. some provider's root servers). This setting can be superseded be a per connection setting as all other `OPENVPN_DEFAULT_` settings can.

OPENVPN_DEFAULT_KEYSIZE Default: `OPENVPN_DEFAULT_KEYSIZE="`

Keysize depends on the encryption method used. Only change this setting when connecting to an OpenVPN remote station that does not use default settings and which you have no influence on. If keysize can be determined by you this value should stay empty. OpenVPN will use the optimal keysize for the encryption method used then.

OPENVPN_DEFAULT_OPEN_OVPNPORT Default: `OPENVPN_DEFAULT_OPEN_OVPNPORT='yes'`

fli4l's packet filter rules have to be changed to enable OpenVPN connections. For all TCP or UDP ports (see `OPENVPN_x_PROTOCOL`) OpenVPN should listen on PF_INPUT_x (Page 40) in base.txt has to be adapted. By specifiying `'yes'` these packet filter rules will be generated automatically. For some connections it may make sense to set `'no'` and define the rules yourself.

OPENVPN_DEFAULT_ALLOW_ICMPPING Default: `OPENVPN_DEFAULT_ALLOW_ICMPPING='yes'`

`'yes'` configures the packet filter for the connection to let pass ping data packets. If there is no really good cause ICMP ping should be allowed at any time. This setting is *not* equivalent to OpenVPN's ping option!

OPENVPN_DEFAULT_PF_INPUT_LOG Default: `OPENVPN_DEFAULT_PF_INPUT_LOG='BASE'`

`'yes'` or `'no'` sets whether the packet filter should protocol denied incoming packets for the VPN connection in the INPUT list or not. By specifiying `'BASE'` the setting from `'PF_INPUT_LOG='` in base.txt will be used.

OPENVPN_DEFAULT_PF_INPUT_POLICY Default: `OPENVPN_DEFAULT_PF_INPUT_POLICY='REJECT'`

This setting equals `'PF_INPUT_POLICY='` (Page 51) in base.txt. By specifiying `'BASE'` the setting from `'PF_INPUT_POLICY='` in base.txt will be used.

OPENVPN_DEFAULT_PF_FORWARD_LOG Default: `OPENVPN_DEFAULT_PF_FORWARD_LOG='BASE'`

`'yes'` or `'no'` sets whether the packet filter should protocol denied incoming packets for the VPN connection in the FORWARD list or not. By specifiying `'BASE'` the setting from `'PF_FORWARD_LOG='` in base.txt will be used.

OPENVPN_DEFAULT_PF_FORWARD_POLICY Default: `OPENVPN_DEFAULT_PF_FORWARD_POLICY='REJECT'`

This setting equals `'PF_FORWARD_POLICY='` (Page 52) in base.txt. By specifiying `'BASE'` the setting from `'PF_FORWARD_POLICY='` in base.txt will be used.

OPENVPN_DEFAULT_PING Default: `OPENVPN_DEFAULT_PING='60'`

To keep an established tunnel open and to recognize if the OpenVPN remote station can still be reached an encrypted ping will be sent over the line in the interval in seconds specified here. `'off'` does not send pings over the line but only real user data.

OPENVPN_DEFAULT_PING_RESTART Default: `OPENVPN_DEFAULT_PING_RESTART='180'`

If in the time interval set here no ping or other data is transferred succesfully the VPN connection concerned will be restarted. The value in `OPENVPN_DEFAULT_PING_RESTART` has to be greater than the one in `OPENVPN_DEFAULT_PING`. `'off'` disables automatic restart.

OPENVPN_DEFAULT_RESOLV_RETRY Default: `OPENVPN_DEFAULT_RESOLV_RETRY='infinite'`

If `OPENVPN_x_REMOTE_HOST` or `OPENVPN_x_LOCAL_HOST` holds DNS names instead of IP addresses they have to be resolved to IP addresses when starting an OpenVPN connection. If this fails OpenVPN will retry to resolve the DNS name for the timespan set here. If this doesn't work within the time limit set here no OpenVPN connection will be established. With `'infinite'` OpenVPN will try forever to resolve the DNS name. Only change this setting if if you know what you're doing!

OPENVPN_DEFAULT_RESTART Default: `OPENVPN_DEFAULT_RESTART='ip-up'`

After disconnection of a tunnel an immedeate restart should be done in order to keep disconnection time as small as possible. For all OpenVPN connections made over dial-in lines like DSL or ISDN `'ip-up'` should be specified here. `'never'` should be set instead for OpenVPN connections over WLAN because of reconnection being independent of dial-ins. For OpenVPN tunnels over an ISDN dial-in connection being established with `ISDN_CIRC_x_TYPE='raw' 'raw-up'` has to be set here.

OPENVPN_DEFAULT_PROTOCOL Default: `OPENVPN_DEFAULT_PROTOCOL='udp'`

This variable sets which protocol should be used as default. UDP is a a good choice normally but sometimes only TCP is allowed, which has a remarkable overhead. Possible values are `'udp'`, `'udp6'`, `'tcp-server'`, `'tcp-server6'`, `'tcp-client'` or `'tcp-client6'`. Settings `'tcp-server'` or `'tcp-client'` make only sense if a VPN tunnel has to be established through a number of paket filters or other tunnels. If no special case should be handled *always* use the default setting `'udp'`. By adding '6' the tunnel will be IPv6 capable (WAN) and can be reached over IPv6-Internet.

OPENVPN_DEFAULT_START Default: `OPENVPN_DEFAULT_START='always'`

OpenVPN connections can either be started `'always'` or `'on-demand'`. Particular OpenVPN connections can be started with the OpenVPN WebGUI (see 4.14.6) only when needed. They can also be started via fli4l console at any time. Login to the fli4l console and execute the following command:

```
cd /etc/openvpn
openvpn --config name.conf --daemon openvpn-name
```

This start an OpenVPN tunnel running in background. Instead of `name.conf` use the name of your configuration file in directory `/etc/openvpn`.

OPENVPN_DEFAULT_VERBOSE Default: `OPENVPN_DEFAULT_VERBOSE='2'`

This variable sets the verbosity of OpenVPN. If a VPN connection is running flawlessly you can set this to `'0'` to avoid all messages. For testing purposes a value of `'3'` is adviced. Higher values may be useful for debugging. Maximum value is `'11'`.

OPENVPN_DEFAULT_MANAGEMENT_LOG_CACHE Default:
`OPENVPN_DEFAULT_MANAGEMENT_LOG_CACHE='100'`

This value controls how many log lines should be saved. Logs can be reviewed in the WebGUI (Page 175).

OPENVPN_DEFAULT_MUTE_REPLAY_WARNINGS Default:
`OPENVPN_DEFAULT_MUTE_REPLAY_WARNINGS='no'`

This variable controls if a warning is posted to the log when receiving double packet for this could point out security problems in a network. When using poor WLAN connections doubled packets may occur rather often. In this case it makes sense to switch the warning off to avoid flooding logfiles. This setting has *no* impact on the security of an OpenVPN connection.

OPENVPN_DEFAULT_MSSFIX Default: `OPENVPN_DEFAULT_MSSFIX="`

Setting MSSFIX defines the size of TCP packets for the VPN connection. `OPENVPN_DEFAULT_MSSFIX='0'` disables this option. If fragment sizes are given and MSSFIX entry is empty fragment sizes will be used automatically. This setting only works with `OPENVPN_x_PROTOCOL='udp'`.

OPENVPN_DEFAULT_FRAGMENT Default: `OPENVPN_DEFAULT_FRAGMENT='1300'`

Activates internal fragmentation of OpenVPN packets with a size of x bytes. This setting only works with `OPENVPN_x_PROTOCOL='udp'`.

`OPENVPN_DEFAULT_FRAGMENT='0'` completely deactivates fragmentation.

OPENVPN_DEFAULT_TUN_MTU Default: `OPENVPN_DEFAULT_TUN_MTU='1500'`

Sets the MTU of the virtual OpenVPN adapter to x bytes. Only change this setting if if you know what you're doing! Usually it is more reasonable to try fragment or MSSFIX options at first.

OPENVPN_DEFAULT_TUN_MTU_EXTRA Default: `OPENVPN_DEFAULT_TUN_MTU_EXTRA="`

If `OPENVPN_x_PROTOCOL='bridge'` is set 32 bytes will be reserved as extra memory for managing the buffers for the tap device. With `OPENVPN_x_PROTOCOL='tunnel'` no extra memory is reserved. This only affects the memory footprint in the router and has no influence on the amount of data sent over the tunnel.

OPENVPN_DEFAULT_LINK_MTU Default: `OPENVPN_DEFAULT_LINK_MTU="`

Sets the MTU of an OpenVPN connection to x bytes. Only use this setting if if you know what you're doing! Usually it is more reasonable to try fragment or MSSFIX options at first.

OPENVPN_DEFAULT_SHAPER Default: `OPENVPN_DEFAULT_SHAPER="`

Restricts *outgoing* bandwidth of the tunnel to the specified value of bytes per second. Possible range is from 100 up to 100000000 bytes. For values up to 1000 bytes per second reduce MTU of the connection otherwise ping times will increase significantly. If you want to restrict a tunnel to a certain bandwidth in both directions you have to configure this option on both OpenVPN end points separately.

In modern OpenVPN versions shaping is not working correctly. Data transfer rates in tunnels using shaping may be extremely fluctuating or even not work at all. Problems may occur in completely different ways depending on the hardware used and lead to unpredictable behavior. Please use shaping with care at the moment. If in doubt deactivate or at least test shaping extensively.

OPENVPN_EXPERT Default: `OPENVPN_EXPERT='no'`

Expert mode enables you to use native Openvpn config files. These have to be stored in the config directory etc/openvpn. All files found there will be transferred to the router.

Expert mode ignores all config settings thus `OPENVPN_N='0'` has to be set.

Expert mode creates no firewall rules. You will have to place them in base.txt by yourself.

Connection-specific Settings

The following OpenVPN options only are valid for the connection mentioned. Only a few of them are mandatory while the most can be omitted. All default settings are taken from `OPENVPN_DEFAULT_x`. Changing values in `OPENVPN_DEFAULT_` applies to all connections that do not explicitely change defaults.

OPENVPN_x_NAME Default: `OPENVPN_x_NAME="`

Defines a name for the OpenVPN connection with up to 16 characters. A config file with this name and suffix .conf will be created in directory /etc/openvpn. This name will appear in syslogs as well. Example: if the name `'peter'` is entered in syslog the connection will appear as 'openvpn-peter'. This helps to identify connections. A name may contain characters, numbers and the '-'.

OPENVPN_x_ACTIV Default: `OPENVPN_x_ACTIV='yes'`

If you want to deactivate an OpenVPN connection but keep the config file it can be disabled by specifying 'no'. Config files will be written to rc.cfg but no corresponding connection will be created.

OPENVPN_x_CHECK_CONFIG Default: `OPENVPN_x_CHECK_CONFIG='yes'`

OpenVPN's extended config file checks are too stringent in rare cases. For example if an ISDN backup connection uses the same routing entries as a connection over the Internet extended checks will complain. In this case extended checking should be disabled for the backup connection. Set `OPENVPN_x_CHECK_CONFIG='no'` to switch off extended checking for this connection.

OPENVPN_x_CIPHER Default see: `OPENVPN_DEFAULT_CIPHER`

See `OPENVPN_DEFAULT_CIPHER` (Page 167). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_COMPRESS Default see: OPENVPN_DEFAULT_COMPRESS

See OPENVPN_DEFAULT_COMPRESS (Page 167). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_CREATE_SECRET Default see: OPENVPN_DEFAULT_CREATE_SECRET='no'

See OPENVPN_x_SECRET (Page 162). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_DIGEST Default see: OPENVPN_DEFAULT_DIGEST

See OPENVPN_DEFAULT_DIGEST (Page 168). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_FLOAT Default see: OPENVPN_DEFAULT_FLOAT

See OPENVPN_DEFAULT_FLOAT (Page 168). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_KEYSIZE Default see: OPENVPN_DEFAULT_KEYSIZE

See OPENVPN_DEFAULT_KEYSIZE (Page 168). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_ISDN_CIRC_NAME Default OPENVPN_x_ISDN_CIRC_NAME=”

Specifies on which ISDN circuit the OpenVPN connection will be established. Enter the name of the ISDN circuits defined in ISDN_CIRC_x_NAME=" (Page 149). The ISDN Circuit has to be of type 'raw'.

OPENVPN_x_PING Default see: OPENVPN_DEFAULT_PING

See OPENVPN_DEFAULT_PING (Page 169). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_PROTOCOL Default: OPENVPN_x_PROTOCOL='udp'

Specifies the protocol to be used for establishing an OpenVPN tunnel. Possible values are 'udp', 'udp6', 'tcp-server', 'tcp-server6', 'tcp-client' or 'tcp-client6'. Settings 'tcp-server' or 'tcp-client' make only sense if a VPN tunnel has to be established through a number of paket filters or other tunnels. If no special case should be handled *always* use the default setting 'udp'. By adding '6' the tunnel will be IPv6 capable (WAN) and can be reached over IPv6-Internet.

OPENVPN_x_RESOLV_RETRY Default see: OPENVPN_DEFAULT_RESOLV_RETRY

See OPENVPN_DEFAULT_RESOLV_RETRY (Page 169). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_PING_RESTART Default see: OPENVPN_DEFAULT_PING_RESTART

See OPENVPN_DEFAULT_PING_RESTART (Page 169). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_START Default see: OPENVPN_DEFAULT_START

See OPENVPN_DEFAULT_START (Page 169). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_VERBOSE Default see: OPENVPN_DEFAULT_VERBOSE

See OPENVPN_DEFAULT_VERBOSE (Page 170). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_MANAGEMENT_LOG_CACHE Default see: OPENVPN_DEFAULT_MANAGEMENT_LOG_CACHE

See OPENVPN_DEFAULT_MANAGEMENT_LOG_CACHE (Page 170). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_MUTE_REPLAY_WARNINGS Default see: OPENVPN_DEFAULT_MUTE_REPLAY_WARNINGS

See OPENVPN_DEFAULT_MUTE_REPLAY_WARNINGS (Page 170). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_RESTART Default see: OPENVPN_DEFAULT_RESTART

See OPENVPN_DEFAULT_RESTART (Page 169). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_ALLOW_ICMPPING Default see: OPENVPN_DEFAULT_ALLOW_ICMPPING

See OPENVPN_DEFAULT_ALLOW_ICMPPING (Page 168). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_OPEN_OVPNPORT Default see: OPENVPN_DEFAULT_OPEN_OVPNPORT

See OPENVPN_DEFAULT_OPEN_OVPNPORT (Page 168). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_PF_INPUT_LOG Default see: OPENVPN_DEFAULT_PF_INPUT_LOG

See OPENVPN_DEFAULT_PF_INPUT_LOG (Page 168). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_PF_INPUT_POLICY Default see: OPENVPN_DEFAULT_PF_INPUT_POLICY

See OPENVPN_DEFAULT_PF_INPUT_POLICY (Page 168). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_PF_INPUT_N Default: OPENVPN_x_PF_INPUT_N='0'

Sets the count of the following OPENVPN_x_PF_INPUT_x= entries.

OPENVPN_x_PF_INPUT_x Default: OPENVPN_x_PF_INPUT_x="

Like in package base this variables contain the packet filter rules. The same syntax like in base.txt is used, tmpl: and host aliased are possible as well. In addition you can use some special symbolic names which are:

VPNDEV actual tun device of the der respective OpenVPN connection.

LOCAL-VPN-IP Uses IP addresses from OPENVPN_x_LOCAL_VPN_IP.

REMOTE-VPN-IP Uses IP addresses from OPENVPN_x_REMOTE_VPN_IP.

REMOTE-NET Uses IP addresses from OPENVPN_x_REMOTE_VPN_IP and in addition all nets specified in OPENVPN_x_ROUTE_x.

OPENVPN_x_PF_FORWARD_LOG Default see: OPENVPN_DEFAULT_PF_FORWARD_LOG

See OPENVPN_DEFAULT_PF_FORWARD_LOG (Page 168). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_PF_FORWARD_POLICY Default see: OPENVPN_DEFAULT_PF_FORWARD_POLICY

See OPENVPN_DEFAULT_PF_FORWARD_POLICY (Page 168). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_PF_FORWARD_N Default: OPENVPN_x_PF_FORWARD_N='0'

Holds the count of the following OPENVPN_x_PF_FORWARD_x= entries.

OPENVPN_x_PF_FORWARD_x Default: OPENVPN_x_PF_FORWARD_x="

See OPENVPN_x_PF_INPUT_x (Page 173).

OPENVPN_x_PF_PREROUTING_N Default: OPENVPN_x_PF_PREROUTING_N='0'

Holds the count of the following OPENVPN_x_PF_PREROUTING_x= entries.

OPENVPN_x_PF_PREROUTING_x Default: OPENVPN_x_PF_PREROUTING_x="

See OPENVPN_x_PF_INPUT_x (Page 173).

OPENVPN_x_PF_POSTROUTING_N Default: OPENVPN_x_PF_POSTROUTING_N='0'

Holds the count of the following OPENVPN_x_PF_POSTROUTING_x= entries.

OPENVPN_x_PF_POSTROUTING_x Default: OPENVPN_x_PF_POSTROUTING_x="

As of fli4l version 3.5.0 (or 3.5.0-rev18133 for tarball users) behavior has changed here. Prior to this entries like

```
OPENVPN_1_PF_POSTROUTING_1='MASQUERADE'
```

were valid. As of now giving a source and a target address is mandatory. This was necessary to use the full extent of POSTROUTING rules. In most cases you will only have to adapt rules IP_NET_x (Page 38) and REMOTE-NET.

See OPENVPN_x_PF_INPUT_x (Page 173).

OPENVPN_x_PF6_INPUT_N Default: OPENVPN_x_PF6_INPUT_N='0'

Holds the count of the following OPENVPN_x_PF6_INPUT_x= entries.

OPENVPN_x_PF6_INPUT_x Default: OPENVPN_x_PF6_INPUT_x="

Here the packet rules have to be set like in package IPv6. Syntax is the same as in ipv6.txt. Also tmpl: and host aliases are possible. In addition you can use some special symbolic names. See OPENVPN_x_PF_INPUT_x (Page 173) for details.

OPENVPN_x_PF6_FORWARD_N Default: OPENVPN_x_PF6_FORWARD_N='0'

Holds the count of the following OPENVPN_x_PF6_FORWARD_x= entries.

OPENVPN_x_PF6_FORWARD_x Default: OPENVPN_x_PF6_FORWARD_x="

See OPENVPN_x_PF6_INPUT_x (Page 174).

OPENVPN_x_MSSFIX Default see: OPENVPN_DEFAULT_MSSFIX

See OPENVPN_DEFAULT_MSSFIX (Page 170). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_FRAGMENT Default see: OPENVPN_DEFAULT_FRAGMENT

See OPENVPN_DEFAULT_FRAGMENT (Page 170). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_TUN_MTU Default see: OPENVPN_DEFAULT_TUN_MTU

See OPENVPN_DEFAULT_TUN_MTU (Page 170). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_TUN_MTU_EXTRA Default see: OPENVPN_DEFAULT_TUN_MTU_EXTRA

See OPENVPN_DEFAULT_TUN_MTU_EXTRA (Page 170). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_LINK_MTU Default see: OPENVPN_DEFAULT_LINK_MTU

See OPENVPN_DEFAULT_LINK_MTU (Page 170). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

OPENVPN_x_SHAPER Default see: OPENVPN_DEFAULT_SHAPER="

See OPENVPN_DEFAULT_SHAPER (Page 171). In contradiction to the default setting this setting only affects the OpenVPN connection mentioned.

4.14.6. OpenVPN - WebGUI

As of fli4l version 2.1.10 a WebGUI is present to start and stop OpenVPN connections and for some more basic functions. You will need to activate package mini_httpd. If you set variable OPENVPN_WEBGUI in openvpn.txt to 'yes' a menu for OpenVPN will be added to the web interface. An overview of the configured connections is displayed when selecting it along with the state and actions possible. (see figure 4.3).

OpenVPN - WebGUI - Connection Overview

Status: State of a connection is symbolized by traffic light symbols. A red man means no OpenVPN process is running a yellow man means the process is running but no connection could be established (yet) and a green man means that a remote station is „connected“. Details about the connection can be displayed as tool tips for the men. This can be of interest at least for „yellow“ status.

Name: In this column the name of the OpenVPN connection is displayed as defined in configuration. A click on the name leads to an overview showing more informations concerning the connection. See below for details.

Aktion: The actions provided are symbolized by buttons with a small definition of each realized by a tool tip. These buttons exist:

Figure 4.3.: Connection Overview

Symbol	Description
	restart OpenVPN process and try to connect.
	stop OpenVPN process.
	reset connection.
	reset connection and put it in 'hold'. No data can be transferrred.
	free connection again. Data can be transferrred.

Table 4.9.: Actions of the OpenVPN-Webgui

OpenVPN - WebGUI - Detail View Of A Connection

Statistics: Some interesting statistics are shown here if the connection is started and not on 'hold'.

Figure 4.4.: Detail view of a connection (Keymanagement)

Log: last 20 lines of the connection logfile. If more lines should be displayed enter the number and click 'show'. If 'all' is selected the whole logfile will be shown. This tab is only shown for started connections.

Debug-Log: Shows the output of the start process. OpenVPN connections are started and the output is shown. This is handy if the connection can't be started by the start button and no normal log can be shown. This tab is only shown for connections not started.

Packet filter: Shows the configuration of the the packet filter relevant to this connection. Packet filter is only configured if the connection is started and configured as a tunnel.

Bridge: Shows the configuration of bridges on the router. Tab is only shown if the connection is configured as a bridge.

Configuration: Shows the configuration generated on boot for this connection.

Keymanagement: Creates a key for the connection and makes it available for download (see figure 4.4). If no key is present on first start it will be generated and displayed automatically. You may download it via the corresponding symbol or copy/paste it to a text file. To save a newly generated keyfile on the router click the disk symbol. Saving can be undone by clicking the restore symbol.

Support informations: Shows all informations relevant when problems occur. You may copy&paste these informations i.e. for a post on the newsgroups.

4.14.7. OpenVPN - Collaboration Of Different OpenVPN Versions

Please note that different versions of OpenVPN may use different default parameters for a connection. In particular MTU fragment and MSSFIX settings may differ. If values „don't match" connection establishment is not possible or no reliable connection can be made. Typical error messages can be:

```
FRAG_IN error flags=0xfa2a187b: FRAG_TEST not implemented
FRAG_IN error flags=0xfa287f34: spurrious FRAG_WHOLE flags
```

Crucial parameters for a connection are:

OPENVPN_x_TUN_MTU MTU Values of the TUN device were set to 1300 for OpenVPN 1.x. As of OpenVPN 2.0 1500 is the default here.

OPENVPN_x_LINK_MTU Byte size of the connection in both OpenVPN daemons. The default is depending on OpenVPN Version and operating system version.

OPENVPN_x_FRAGMENT Data packets (UDP or TCP) with a size bigger than the fragment size will be fragmented to packets not bigger than byte size provided in `OPENVPN_x_FRAGMENT`.

OPENVPN_x_MSSFIX To avoid fragmentation of data packets for TCP connections over VPN a maximum size for TCp data packets can be set here. Up-to-date operating systems will honorate this setting and make fragmentation unnecessary.

Different OpenVPN versions use the following settings as default values. Please obey these values when connecting OpenVPN in varying versions. Default settings on fli4l routers are shown in the second table.

OpenVPN Version/Option	1.xx	2.00
OPENVPN_x_TUN_MTU	1300	1500
OPENVPN_x_TUN_MTU_EXTRA	unknown	32
OPENVPN_x_FRAGMENT	unknown	not configured
OPENVPN_x_MSSFIX	not configured	1450

Table 4.10.: Different MTU parameters in different OpenVPN versions.

fli4l Version/Option	up to 2.1.8	from 2.1.9 on
OPENVPN_x_TUN_MTU	1300	1500
OPENVPN_x_TUN_MTU_EXTRA	64	32
OPENVPN_x_FRAGMENT	not configured	1300
OPENVPN_x_MSSFIX	not configured	1300

Table 4.11.: Different MTU parameters in different fli4l router versions.

Based on this settings the defaults for your network should be determined and written to config/openvpn.txt explicitely. These are the best values for your tests to start with:

```
OPENVPN_DEFAULT_TUN_MTU='1500'
OPENVPN_DEFAULT_MSSFIX='1300'
OPENVPN_DEFAULT_FRAGMENT='1300'
```

For fli4l versions prior to 2.1.9 „tun-mtu“ parameters can't be specified directly. But they can be influenced indirectly with `OPENVPN_x_LINK_MTU`. tun-mtu values are about 45 byte smaller than the values in `OPENVPN_x_LINK_MTU`. To get exact values only trying will help.

4.14.8. OpenVPN - Examples

Some examples will clarify the configuration of package OpenVPN.

Example - Joining Two Nets Using fli4l Routers

In the first example two fli4l routers will be connected. Nets behind each fli4l router should gain access to each other. Peter and Maria want to connect their nets over their fli4l routers. Peter uses a private net 192.168.145.0/24 and a DynDNS address 'peter.eisfair.net'. Marias setup is similar while she is using 10.23.17.0/24 and DynDNS address 'maria.eisfair.net'. Both trust in each other so they allow unlimited access to their complete nets for each other.

OpenVPN Option	Peter	Maria
OPENVPN_1_NAME=	'maria'	'peter'
OPENVPN_1_REMOTE_HOST=	'maria.eisfair.net'	'peter.eisfair.net'
OPENVPN_1_REMOTE_PORT=	'10000'	'10001'
OPENVPN_1_LOCAL_PORT=	'10001'	'10000'
OPENVPN_1_SECRET=	'pema.secret'	'pema.secret'
OPENVPN_1_TYPE=	'tunnel'	'tunnel'
OPENVPN_1_REMOTE_VPN_IP=	'192.168.200.202'	'192.168.200.193'
OPENVPN_1_LOCAL_VPN_IP=	'192.168.200.193'	'192.168.200.202'
OPENVPN_1_ROUTE_N=	'1'	'1'
OPENVPN_1_ROUTE_1=	'10.23.17.0/24'	'192.168.145.0/24'
OPENVPN_1_PF_INPUT_N=	'1'	'1'
OPENVPN_1_PF_INPUT_1=	'ACCEPT'	'ACCEPT'
OPENVPN_1_PF_FORWARD_N=	'1'	'1'
OPENVPN_1_PF_FORWARD_1=	'ACCEPT'	'ACCEPT'

Table 4.12.: OpenVPN Configuration with 2 fli4l routers

Example - Two Nets Connected By A Bridge

In the next example a bridge over a wi-fi connection will be configured. Packet filters are not of use here because usually ethernet frames will be forwarded but no IP packets. Please remember that with a bridge a common net is used. Thus no IP address can exist twice.

In addition to the settings for OpenVPN a bridge has to be configured in advanced_networking and base.txt has to be adapted to use the bridge device and not eth0 as the network device for the internal net. See the relevant entries in advanced_networking's and base's configuration files:

Example - Configure Access For A Road Warrior

For this example (Roadwarrior) access to a LAN behind fli4l should be configured for a notebook with Windows XP over GPRS. OpenVPN is installed on the notebook and the *.ovpn file

OpenVPN Option	Peter	Maria
OPENVPN_2_NAME	'bridge'	'bridge'
OPENVPN_2_REMOTE_HOST	'10.1.0.1'	'10.2.0.1'
OPENVPN_2_REMOTE_PORT	'10005'	'10006'
OPENVPN_2_LOCAL_HOST	'10.2.0.1'	'10.1.0.1'
OPENVPN_2_LOCAL_PORT	'10006'	'10005'
OPENVPN_2_FLOAT	'no'	'no'
OPENVPN_2_RESTART	'never'	'never'
OPENVPN_2_SECRET	'bridge.secret'	'bridge.secret'
OPENVPN_2_TYPE	'bridge'	'bridge'
OPENVPN_2_BRIDGE	'pema-br'	'pema-br'

Table 4.13.: OpenVPN Configuration with 2 fli4l routers bridging nets over a wi-fi connection.

advanced_networking Option	Peter	Maria
OPT_BRIDGE_DEV	'yes'	'yes'
BRIDGE_DEV_BOOTDELAY	'no'	'no'
BRIDGE_DEV_N	'1'	'1'
BRIDGE_DEV_1_NAME	'pema-br'	'pema-br'
BRIDGE_DEV_1_DEVNAME	'br0'	'br0'
BRIDGE_DEV_1_DEV_N	'1'	'1'
BRIDGE_DEV_1_DEV_1_DEV	'eth0'	'eth0'

Table 4.14.: OpenVPN Configuration with 2 fli4l routers bridging nets over a wi-fi connection. Bridge configuration in advanced_networking.

base Option	Peter	Maria
IP_NET_N	'1'	'1'
IP_NET_1	'192.168.193.254/24'	'192.168.193.1/24'
IP_NET_1_DEV	'br0'	'br0'

Table 4.15.: OpenVPN Configuration with 2 fli4l routers bridging nets over a wi-fi connection. Bridge configuration in base.txt.

is edited. Unfortunately the tun/tap driver for Windows is not as flexible as its Unix pendant. Point-to-Point addresses for VPN IP have to be in a 255.255.255.252 (or /30) net. If the road warrior should only access services in the LAN behind or on the fli4l router itself and does not have to be accessed by itself a route on fli4l's side is not necessary. The road warrior can be addressed on its virtual IP address (`OPENVPN_3_REMOTE_VPN_IP`) if necessary. If the road warrior has a fixed IP address a host route could be added if needed. If the road warrior i.e. has fixed IP address 192.168.33.33 you could simply add the following to fli4l's openvpn.txt:

```
OPENVPN_3_ROUTE_N='1'
OPENVPN_3_ROUTE_1='192.168.33.33/32'
```

With the configuration of the packet filter shown here complete communication in both directions is allowed. Only the fli4l router is not directly accessible for the road warrior. That would be needed if the road warrior should use the DNS server on the fli4l router.

```
OPENVPN_3_PF_FORWARD_N='1'
OPENVPN_3_PF_FORWARD_1='ACCEPT'
```

For allowing access to fli4l's internal DNS server add the following to the configuration of fli4l:

```
OPENVPN_3_PF_INPUT_N='1'
OPENVPN_3_PF_INPUT_1='if:VPNDEV:any tmpl:dns ACCEPT'
```

OpenVPN Option fli4l router	roadwarrior
OPENVPN_3_NAME='roadwarrior'	remote peter.eisfair.net
OPENVPN_3_LOCAL_PORT='10011'	rport 10011
OPENVPN_3_SECRET='roadwarrior.secret'	secret roadwarrior.secret
OPENVPN_3_TYPE='tunnel'	dev tun
OPENVPN_3_REMOTE_VPN_IP='192.168.200.238'	
OPENVPN_3_LOCAL_VPN_IP='192.168.200.237'	ifconfig 192.168.200.238 192.168.200.237
OPENVPN_3_ROUTE_N='0'	
OPENVPN_3_PF_FORWARD_N='1'	
OPENVPN_3_PF_FORWARD_1='ACCEPT'	
	route 192.168.145.0 255.255.255.0
	comp-lzo
	persist-tun
	persist-key
	ping-timer-rem
	ping-restart 60
	proto udp
	tun-mtu 1500
	fragment 1300
	mssfix

Table 4.16.: OpenVPN Configuration with a Windows Computer over GPRS.

Example - Secure A WLAN Connection

In this example a WLAN connection will be secured by the help of OpenVPN. The fli4l router has a LAN and a WLAN card it uses or an access point is connected to an additional fli4l NIC. This aims at WLAN clients only having access to the VPN port without establishing a VPN connection. After connecting succesfully to OpenVPN they should have unlimited access with

cable nets. DNSMASQ DHCP server's settings have to be changed to achieve that. Package advanced_networking will be needed as well. Settings in base.txt: `IP_NET_1` is the cable LAN and `IP_NET_2` is the WLAN.

```
IP_NET_N='2'
IP_NET_1='192.168.3.254/24'
IP_NET_1_DEV='br0'
IP_NET_2='192.168.4.254/24'
IP_NET_2_DEV='eth2'
```

Set DHCP range to suit your needs. For `IP_NET_2` two settings are mandatory:

```
DHCP_RANGE_2_DNS_SERVER1='none'
DHCP_RANGE_2_NTP_SERVER='none'
DHCP_RANGE_2_GATEWAY='none'
```

Settings in advanced_networking.txt:

```
OPT_BRIDGE_DEV='yes'
BRIDGE_DEV_BOOTDELAY='yes'
BRIDGE_DEV_N='1'
BRIDGE_DEV_1_NAME='br'
BRIDGE_DEV_1_DEVNAME='br0'
BRIDGE_DEV_1_DEV_N='1'
BRIDGE_DEV_1_DEV_1_DEV='eth0'
```

OpenVPN Option Router	WLAN-Client
OPENVPN_4_NAME='wlan1'	
OPENVPN_4_LOCAL_HOST='192.168.4.254'	remote 192.168.4.254
OPENVPN_4_LOCAL_PORT='20001'	rport 20001
OPENVPN_4_SECRET='wlan1.secret'	secret wlan1.secret
OPENVPN_4_TYPE='bridge'	dev tap
OPENVPN_4_BRIDGE='br'	
OPENVPN_4_RESTART='never'	
OPENVPN_4_MUTE_REPLAY_WARNINGS='yes'	
	comp-lzo
	persist-tun
	persist-key
	ping-timer-rem
	ping-restart 60
	proto udp
	tun-mtu 1500
	fragment 1300
	mssfix

Table 4.17.: OpenVPN Securing a WLAN.

4.14.9. Additional Links On OpenVPN

At the end here are some links on OpenVPN configuration:
http://openvpn.net
http://de.wikipedia.org/wiki/OpenVPN
http://openvpn.se/
http://arnowelzel.de/wiki/de/fli4l/openvpn

http://wiki.freifunk.net/OpenVPN
http://w3.linux-magazine.com/issue/24/Charly.pdf
http://w3.linux-magazine.com/issue/25/WirelessLAN_Intro.pdf
http://w3.linux-magazine.com/issue/25/OpenVPN.pdf

4.15. PCMCIA - PC-Card Support

4.15.1. PCMCIA Drivers

fli4l can work with PCMCIA cards as well. Specify OPT_PCMCIA='yes' to install according base drivers. The card driver to be used concretely is set by NET_DRV_x (Page 32).

PCMCIA_PCIC - PCMCIA socket driver

It can be chosen between: 'i82365' or 'tcic' for PCMCIA bridges and 'yenta_socket' res. 'i82092' for Cardbus bridges.

Default setting: PCMCIA_PCIC='i82365'

PCMCIA_PCIC_OPTS - Options for the PCMCIA socket driver

Default setting: PCMCIA_PCIC_OPTS=''

Possible Settings:
poll_interval=n n in intervals of 10 milliseconds - reasonable value: 1000 Sets polling interval for card changes.
irq_list=x,y,z,... A list of interrupts to be used

PCMCIA_MISC_N PCMCIA_MISC_x Number of PCMCIA modules to be loaded additionally: serial_cs for modems and combo cards parport_cs printer interface cards

4.16. PPP - Connection Of Computers Via Serial Interface

With OPT_PPP='yes' a computer can be connected via its serial interface. This may be useful i.e. for connecting computers to a network that does not have a network interface card. The computer connected to the serial interface is called client PC below.

PPP_DEV Specify fli4l's serial interface the client PC is connected to. Possible values are:

'com1' COM1 port (lower cases only!)
'com2' COM2 port (lower cases only!)

PPP_SPEED Set transfer rate here (bit/sec). 38400 is supported by older interfaces too. Problems may occur when using rates above this.

```
Example: PPP_SPEED='38400'
```

PPP_IPADDR PPP_PEER PPP_IPADDR holds fli4l's IP address on the COM port, i.e. '192.168.4.1'. In variable PPP_PEER the IP address of the client PC is set, i.e. '192.168.4.2'.

PPP_NETWORK PPP_NETMASK PPP_NETWORK holds the network used and variable PPP_NETMASK the netmask. These two variables are used by the extra package 'samba_lpd'.

Important: *Consider the following:*

1. Those IP addresses may **not** originate from the network address range of the ethernet LANs. Point-To-Point-Configuration must use a separate network address range!
2. For the Client PC's connection to the Internet this mini-PPP-network has to be masked in the same way as the LAN. In order to achieve this extend the list of networks to be masked via the variable PF_POSTROUTING_% (Page 56) (see next passage).
3. The client PC should be added to the host table of the DNS server.

Causes:

If telnet or ftp should be used from the client PC to the fli4l router the daemons concerned on fli4l do a reverse DNS lookup to resolve the client PC. If it is not found in the host table a connection to the Internet will be established to do this - which is of course useless. To avoid this enter the client PC in the host table.

Example for PPP Configuration over the serial interface:

```
PPP_DEV='com1'
PPP_SPEED='38400'
PPP_IPADDR='192.168.4.1'
PPP_PEER='192.168.4.2'
PPP_NETWORK='192.168.4.0'
PPP_NETMASK='255.255.255.0'
```

and further on in config/base.txt:

```
PF_POSTROUTING_N='2'
PF_POSTROUTING_1='192.168.6.0/24 MASQUERADE'
PF_POSTROUTING_2='192.168.4.0/24 MASQUERADE'
```

The first network is the one of the ethernet LAN and the second the one of the PPP network.

Last thing is to adapt DNS Configuration. Example:

```
HOST_5='192.168.4.2 serial-pc'
```

Do not forget to increment HOST_N (Page 91)!

If the client PC is running Windows you have to configure the dial-up adapter for a PPP connection to the fli4l router.

If a linux client is used create a shell script on the client (i.e. /usr/local/bin/ppp-on):

```
#! /bin/sh
dev='/dev/ttyS0'                    # COM1, for COM2: ttyS1
speed='38400'                       # speed
options='defaultroute crtscts'      # options
myip='192.168.4.2'                  # IP address client
fli4lip='192.168.4.1'               # IP address fli4l router
pppd $dev $speed $options $myip:$fli4lip &
```

In case of problems: man pppd

The fli4l router has to be used as the DNS server on the client if a connection to the Internet is desired. Add two lines to /etc/resolv.conf on the client: the domain used and the ethernet IP address of the fli4l router as name server.

Example:

```
search domain.de
nameserver 192.168.1.4
```

"domain.de" res. "192.168.1.4" have to be changed to your needs. Important: The IP address has to be the one of fli4l's ethernet card!

A so called null modem cable (Page 338) is used as the physical connection. See appendix for package base for pin wirings.

A (german) Howto for connecting a Windows client with serial PPP can be found at:

http://www.fli4l.de/hilfe/howtos/basteleien/opt-ppp-howto/

4.17. PROXY - Several Proxy Servers

4.17.1. 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 survice 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 187). 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.

4.17.2. 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 authentification 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.

4.17.3. 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`.

4.17.4. 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='tmpl: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.

4.17.5. OPT_SIPPROXY (EXPERIMENTELL) - 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!

4.17.6. 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
  - eht1 is the DSL interface mentioned in dsl.txt

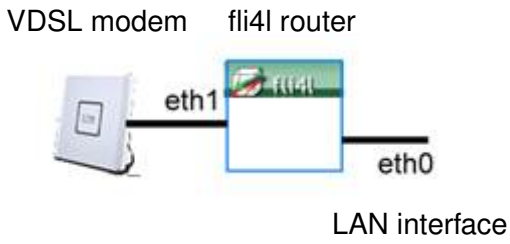


Figure 4.5.: fli4l in a standard configuration

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

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.

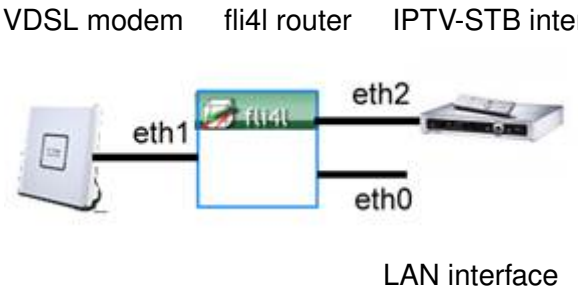


Figure 4.6.: fli4l in an IPTV configuration

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 architecure 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 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 disconnetion 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_NET_x 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_NET_1='239.35.0.0/16'        # IPTV streams - always needed
IGMPPROXY_ALT_NET_2='217.0.119.0/24'       # needed for T-Home
IGMPPROXY_ALT_NET_3='193.158.34.0/23'      # needed for T-Home
                                           # before May 2013 '193.158.35.0/24'
# IGMPPROXY_ALT_NET_4='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_WHLIST_NET_x :

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 goups in this list will be requested on the WAN side.

```
IGMPPROXY_WLIST_N='1'                          # Number of Multicast sources
IGMPPROXY_WHLIST_NET_1='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 exepected 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'
```

4.17.7. 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 whith 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 198).

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 [14]

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 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

[14] see http://en.wikipedia.org/wiki/POP3S

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/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 200).

The certificate must be provided in the so-called PEM format and must be saved below `<config-directory>/etc/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 know.
- *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.

4.18. QoS - Quality of Service

By QoS the available bandwidth can be regulated and for example be distributed to several ports, IP addresses and more.

A modem manages a packet queue where packets are stored that exceed the available bandwidth. With DSL modems for example these queues are rather big. The advantage is a constant usage of maximum bandwidth. If the router sends lesser packets for a short period of time the modem has packets in the queue that it can send. Such a queue is a simple thing sending packets first in first out - rather fair, isn't it?

This is where QoS comes into play. QoS also manages a packet queue in the router itself which makes it possible to decide which packets are to be sent at first and which to hold back. If everything is configured the right way QoS sends packets at the speed the modem sends them out not filling the modem queue at any time. This is like moving the modem queue into the router.

Something general on speed units: QoS supports Mibit/s (mebibit/s) and Kibit/s (kibibit/s), where applies 1Mibit = 1024Kibit.

4.18.1. Configuration

OPT_QOS Set this to 'yes' to enable and 'no' to disable `OPT_QOS`.

QOS_INTERNET_DEV_N Number of devices routing data to the Internet.

QOS_INTERNET_DEV_x List of devices that route data to the Internet. Examples:

`QOS_INTERNET_DEV_N='3'`	device number
`QOS_INTERNET_DEV_1='ethX'`	for cable and other ethernet connections
`QOS_INTERNET_DEV_2='ppp0'`	for DSL over PPPoE
`QOS_INTERNET_DEV_3='ipppX'`	for ISDN

The ISDN device for the first circuit should be named ippp0 the second ippp1. If channel bundeling is activated for the first circuit the second channel of the first circuit is named ippp1 and the second circuit ippp2. QOS should only be used with ISDN if channel bundeling is deactivated for the circuit used.

QOS_INTERNET_BAND_DOWN Maximum downstream bandwidth of the Internet connection. See above: Something general on speed units (Page 203).

Hint: For time-critical jobs like preferring ACK packets it is necessary to limit the bandwidth to the actual value. Else packets will be sorted correctly in the packet queue but are withheld in the modem's packet queue. It may be possible that bandwidth declared by your provider is not in accordance with real conditions. It could be a little less or a little more. At the end only trying helps here.

QOS_INTERNET_BAND_UP Maximum upstream bandwidth of the Internet connection. See above: Something general on speed units (Page 203).

Also see hint at `QOS_INTERNET_BAND_DOWN`.

QOS_INTERNET_DEFAULT_DOWN Set the default class for packets coming from the Internet here. All packets which are not classified by a filter will end here.

If no class is specified for which variable

```
QOS_CLASS_x_DIRECTION='down'
```

is set specify:

```
QOS_INTERNET_DEFAULT_DOWN='0'
```

Example:

Two classes have been created and a filter puts all packets for a certain IP address into the first one. All other packets should go to the second one. This must be set like this:

```
QOS_INTERNET_DEFAULT_DOWN='2'
```

Pay attention to set a class for `QOS_INTERNET_DEFAULT_DOWN` where its `QOS_CLASS_x_DIRECTION` variable contains the argument 'down'.

QOS_INTERNET_DEFAULT_UP Set the default class for packets going out to the Internet here. All packets which are not classified by a filter will end here.

If no class is specified for which variable

```
QOS_CLASS_x_DIRECTION='up'
```

is set specify:

```
QOS_INTERNET_DEFAULT_UP='0'
```

This works in analog to `QOS_INTERNET_DEFAULT_DOWN`.

Pay attention to set a class for `QOS_INTERNET_DEFAULT_UP` where its `QOS_CLASS_x_DIRECTION` variable contains the argument 'up'.

QOS_CLASS_N Set the number of classes to be created.

QOS_CLASS_x_PARENT By this variable classes can be stacked. Set the number of the parent class here. Bandwidth allocated to the parent class can be spread between the subclasses. Maximum subclass layer depth is 8 whereas the interface itself is the first layer which leaves a maximum of 7 layers to be configured.

If the class is no subclass write the following:

```
QOS_CLASS_x_PARENT='0'
```

It will get the maximum bandwidth set in `QOS_CLASS_x_PORT_TYPE` depending on its direction (in- or outbound, see `QOS_CLASS_x_PORT_TYPE`)

Important: If this is not '0' pay attention to define the parent class before (in numbering)

QOS_CLASS_x_MINBANDWIDTH Bandwidth to allocate to the classes. See above: Something general on speed units (Page 203).

Example: A class with a bandwidth limited to 128Kibit/s:

```
QOS_CLASS_1_MINBANDWIDTH='128Kibit/s'
QOS_CLASS_1_MAXBANDWIDTH='128Kibit/s'
QOS_CLASS_1_PARENT='0'
```

Three subclasses of our parent class above where QOS_CLASS_x_MINBANDWIDTH- and QOS_-CLASS_x_MAXBANDWIDTH settings look like this:

```
QOS_CLASS_2_PARENT='1'
QOS_CLASS_2_MINBANDWIDTH='60Kibit/s'
QOS_CLASS_2_MAXBANDWIDTH='128Kibit/s'

QOS_CLASS_3_PARENT='1'
QOS_CLASS_3_MINBANDWIDTH='40Kibit/s'
QOS_CLASS_3_MAXBANDWIDTH='128Kibit/s'

QOS_CLASS_4_PARENT='1'
QOS_CLASS_4_MINBANDWIDTH='28Kibit/s'
QOS_CLASS_4_MAXBANDWIDTH='128Kibit/s'
```

All subclasses have the same (or no) priority (see QOS_CLASS_x_PRIO). If traffic on all subclasses exceeds their QOS_CLASS_x_MINBANDWIDTH they all get QOS_CLASS_x_MINBANDWIDTH allocated. If class 2 has only 20Kibit/s traffic there are 40Kibit/s "left". This overrun will be split in 40/28 ratio between class 3 and 4. Each class is limited to 128Kibit/s by QOS_CLASS_x_MAXBANDWIDTH and because all classes are subclasses of a parent class limited to 128Kibit/s the maximum bandwidth consumed is 128Kibit/s.

QOS_CLASS_x_MAXBANDWIDTH Maximum bandwidth to be allocated to the class. There is no sense in entering a lower value than in QOS_CLASS_x_MINBANDWIDTH. If nothing is set here this variable automatically will be set to the value of QOS_CLASS_x_MINBANDWIDTH. Such a class can't use any free bandwidth resources.

See above: Something general on speed units (Page 203).

QOS_CLASS_x_DIRECTION This variable sets the direction the class belongs to. If upstream regulation is intended set:

```
QOS_CLASS_x_DIRECTION='up'
```

for downstream in analog:

```
QOS_CLASS_x_DIRECTION='down'
```

QOS_CLASS_x_PRIO Set the priority of the class here. The lower the number the higher the priority. Values between 0 and 7 are allowed. Leaving this empty equals to setting '0'.

Priorities are used to determine which class can use free bandwidth resources. Lets change our example in QOS_CLASS_x_MINIMUMBANDWIDTH to reflect this: Nothing was changed for the first class. Classes 2 to 4 get a priority:

```
QOS_CLASS_2_PARENT='1'
QOS_CLASS_2_MINBANDWIDTH='60Kibit/s'
QOS_CLASS_2_MAXBANDWIDTH='128Kibit/s'
```

```
QOS_CLASS_2_PRIO='1'

QOS_CLASS_3_MINBANDWIDTH='40Kibit/s'
QOS_CLASS_3_PARENT='1'
QOS_CLASS_3_MAXBANDWIDTH='128Kibit/s'
QOS_CLASS_3_PRIO='1'

QOS_CLASS_4_PARENT='1'
QOS_CLASS_4_MINBANDWIDTH='28Kibit/s'
QOS_CLASS_4_MAXBANDWIDTH='128Kibit/s'
QOS_CLASS_4_PRIO='2'
```

Like in the original example class 2 consumes only 20Kibit/s and leaves free bandwidth of 40Kibit/s. Classes 3 and 4 both need more bandwidth than available. Class 3 now has a higher priority than class 4 and may use the free bandwidth of 40Kibit/s.

If class 3 needs only 20Kibit/s of the free bandwidth of 40Kibit/s class 4 will get the remaining 20Kibit/s.

Lets assume something different: Class 4 needs no bandwidth at all and class 2 and 3 both need more than exists. So they both get the bandwidth set in `QOS_CLASS_x_MINBANDWIDTH` and the remaining will be divided in 60/40 ratio between them because both classes have the same priority.

As you see `QOS_CLASS_x_PRIO` only has influence on how an enventual overrun in bandwidth is spread.

QOS_CLASS_x_LABEL By using this optional variable a label for the class may be specified which will be used as the caption for the QOS-graphs created by an activated OPT_RRDTOOL.

QOS_FILTER_N Set the number of filters desired here.

A quick note on filters: Arguments of the different variables are AND-linked, arguments of the same variable are OR-linked. This means: If the same filter filters by an IP-address and a port only packets will get selected and put in the queue which match both conditions at a time.

Another example: Two ports (21 and 80) and an IP address are set in the same filter. Since a packet can't use two ports at a time the filter will match those packets that either use port 21 and the named IP address or port 80 and the named IP address.

Important: The ordering of filters is essential!

An example: All traffic on port 456 for **all** clients should be queued to class A. In addition all packets to a client with IP address 192.168.6.5 should be queued to class B, except those on port 456. If the filter on the IP is created first all packets (those on port 456 too) will be queued to class B and the filter on port 456 doesn't change this behavior. The filter on port 456 has to be created before the one on the IP address 192.168.6.5 to achieve our goals.

QOS_FILTER_x_CLASS This variable specifies the class a packet is queued in that matches the given filter. If for example packets should be put in the queue specified by `QOS_CLASS_25_MINBANDWIDTH` this should be done like this:

```
QOS_FILTER_x_CLASS='25'
```

By QOS_CLASS_x_DIRECTION it is set if a class belongs to up- or downstream. If a filter is set then queueing packets to an upstream class only upstream packets will be filtered and queued to the class mentioned. QOS_CLASS_x_DIRECTION defines the "direction " of filtering.

As of version 2.1 more than one target class can be set. If for example traffic on port 456 upstream and downstream is our target

```
QOS_FILTER_x_CLASS='4 25'
```

would be specified, if class 4 is the upstream class and 25 is the downstream one. It does not make sense to specifiy more than one up- or downstream class so there never will be more than two target classes.

QOS_FILTER_x_IP_INTERN Set IP addresses and IP ranges from internal networks here which should be filtered. They have to be separated by spaces and can be combined in any manner.

An example:

```
QOS_FILTER_x_IP_INTERN='192.168.6.0/24 192.168.5.7 192.168.5.12'
```

This filters all IP addresses that match 192.168.6.X and in addition IP 192.168.5.7 and 192.168.5.12.

This variable can be empty.

If this variable is used in conjunction with QOS_FILTER_x_IP_EXTERN only traffic between IPs or IP ranges defined by QOS_FILTER_x_IP_INTERN and QOS_FILTER_x_IP_EXTERN will be filtered.

Important: *If filtering by QOS_FILTER_x_OPTION for ACK, TOSMD, TOSMT, TOSMR or TOSMC takes place and variable QOS_CLASS_x_DIRECTION is of target class 'down' this variable will be ignored.*

QOS_FILTER_x_IP_EXTERN Specify IP addresses and IP ranges from external networks (being connected on QOS_INTERNET_DEV) here which should be filtered. They have to be separated by spaces and can be combined in any manner. This works the same way as QOS_FILTER_x_IP_INTERN.

This variable can be empty.

Important: *If filtering by QOS_FILTER_x_OPTION for ACK, TOSMD, TOSMT, TOSMR or TOSMC takes place and variable QOS_CLASS_x_DIRECTION is of target class 'down' this variable will be ignored.*

QOS_FILTER_x_PORT Ports and port ranges can be set here, separated by spaces and combined in any manner. If this variable is empty traffic on all ports will be limited.

Filtering for a port range from 5000 up to 5099 would look like this:

`QOS_FILTER_x_PORT='5000-5099'`

Another example: If traffic on ports 20 to 21, 137 to 139 and port 80 should be filtered to the same class this would look like this:

```
QOS_FILTER_x_PORT='20-21 137-139 80'
```

This variable can be empty.

Important:

- If filtering for ports `QOS_FILTER_x_PORT_TYPE` has to be set as well.
- Port ranges will be ignored if filtering for ACK, TOSMD, TOSMT, TOSMR or TOSMC takes place by the use of `QOS_FILTER_x_OPTION`.

QOS_FILTER_x_PORT_TYPE This variable is only valid and important in conjunction with `QOS_FILTER_x_PORT` (it is ignored in other cases).

Ports in client mode are different from those in server mode. Specify here if a port is of type server or client. Use PCs from your own net as a point of reference to decide what to use. Possible settings:

```
QOS_FILTER_x_PORT_TYPE='client'
QOS_FILTER_x_PORT_TYPE='server'
```

As of version 2.1 a combination of those two arguments is also valid to put traffic from the own net as well as traffic from the Internet into the same class on this port.

```
QOS_FILTER_x_PORT_TYPE='client server'
```

This equals to two similar filters once with `QOS_FILTER_x_PORT_TYPE` set to client and once set to server.

QOS_FILTER_x_OPTION This variable activates additional properties for the filter. Only one of the following arguments can be passed (a combination wouldn't make sense in the same filter). It is perfectly right and makes sense sometimes to set a filter for ACK packets and a second filter for TOSMD packets to move their packets to the same target class (see `QOS_FILTER_x_CLASS`).

ACK Acknowledgement packets.

A packet matching this option is sent as an acknowledgement to a data packet. If i.e. a huge download is running a lot of data packets will come in and for each of them an acknowledgement has to be sent to confirm it has reached you. If no acknowledgement packets reach the download source it will wait for them before sending the next chunk of data.

This is extremely important with asymetric connections (up- and downstream bandwidths differ) like used in most DSL lines. Those most likely have a small upstream that tends to reach its maximum rather fast. If ACK packets are normally queued this may end in the data server delaying its transaction to wait for ACK packets to come in. This results in download rates lower than they could be.

So ACK packets have to bypass the "normal" packets in order to get to the data server as fast as possible. How to combine this option in a meaningful way with a class is explained in the examples.

ICMP Ping packets (Protocol ICMP)

Ping packets are used to measure time a packet takes to get from A to B. To take influence on this value give ping packets a higher priority. This does not influence ping times for online games.

IGMP IGMP-Pakete (Protokoll IGMP)

If using IP-TV it makes sense to filter and priorize IGMP packets.

TCPSMALL Small TCP Packets

By using this filter outgoing HTTP(s)-Requests can be filtered and priorized. A combination with a destination port is possible and makes sense. Approx. size of this TCP packets is 800 Byte max.

TCP TCP packets (Protocol TCP)

Only packets using protocol TCP are filtered.

UDP UDP packets (Protocol UDP)

Only packets using protocol UDP are filtered.

TOS* Type of Service

An application can set one of four TOS bits for each packet transmitted. This specifies the intended handling for those packets. For example SSH can set TOS-Minimum-Delay for sending in- and output and TOS-Maximum-Troughput for sending files. Linux/Unix programs use these bits more often than Windows programs. A firewall can set TOS bits for certain packets as well. In the end it all depends on routers in the transport chain to honour TOS bits or not. Only TOS bits Minimum-Delay and Maximum-Throughput are really important for fli4l.

TOSMD - TOS Minimum-Delay Used for services that need packets to be transferred without time delays. It is recommended to use this TOS bit for FTP (control data), Telnet and SSH.

TOSMT - TOS Maximum-Troughput Used for services that need big amounts of data to be transferred with high speed. It is recommended to use this TOS bit for FTP data and WWW.

TOSMR - TOS Maximum-Reliability Used to ensure that data reaches its target without being resent. Recommended use for this TOS bit is SNMP and DNS.

TOSMC - TOS Minimum-Cost Used to minimize costs for transferring data. Recommended use for this TOS bit is NNTP and SMTP.

DSCP* Differentiated Services Code Point

DSCP is a marking according to RFC 2474. This process has replaced TOS marking mostly since 1998.

Filters on DSCP-classes can be configured as follows:

```
QOS_FILTER_x_OPTION='DSCPef'
QOS_FILTER_x_OPTION='DSCPcs3'
```

Please note that DSCP is in capital letters while the class is lower case.

The following classes can be used:

af11-af13, af21-af23, af31-af33, af41-af43, cs1-cs7, ef und be (Standard)

4.18.2. Examples

How do we configure `OPT_QoS` in detail? This will be shown below for some use cases:

- Example 1: targets at spreading bandwidth between 3 clients.
- Example 2: targets at spreading bandwidth between 2 clients and on the clients between one port and the rest of the traffic on this client.
- Example 3: targets at learning the general structures of working with QoS.
- Example 4: a configuration for preferring ACK packets in order to keep downstream high even if upstream is overloaded.

Example 1

Spreading bandwidth between 3 clients.

Four classes are created (see `QOS_CLASS_N`) with the following speeds (see `QOS_CLASS_x_MINBANDWIDTH` / `QOS_CLASS_x_MINBANDWIDTH`). They are subclasses of class 0 (see `QOS_CLASS_x_PARENT`) and therefore are bound directly to the interface for "up" res. "down" (see `QOS_CLASS_x_DIRECTION`).

The fourth class is only for visitors and gets lesser bandwidth. By `QOS_INTERNET_DEFAULT_DOWN`='4' all traffic not filtered is queued to the forth "guest" class. Because we seldom have visitors and the bandwidth is the same for the other 3 classes clients get 1/3 of the overall bandwidth (256Kibit/s each).

This configuration is only a beginning. It has to be specified how traffic is regulated by the classes.

We use two filters to assign traffic to the classes. We create 3 filters, one for each of the 3 clients (see `QOS_FILTER_N` a.s.o.) and attach a filter to each class (see `QOS_FILTER_x_CLASS`). By specifying `QOS_FILTER_x_IP_INTERN`, `QOS_FILTER_x_IP_INTERN`, `QOS_FILTER_x_PORT`, `QOS_FILTER_x_PORT` and `QOS_FILTER_x_OPTION` it can be defined what rules apply to each class.

Let's call the interface 0, the 3 classes 1, 2 and 3 and the 3 filters F1, F2 and F3. The scenario looks like this image 4.7.

Configuration looks like this:

Three client PCs filtered by IP with 1/3 bandwidth each if visitors are absent:

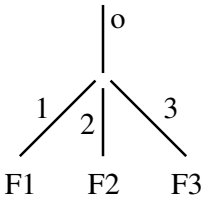


Figure 4.7.: Example 1

```
OPT_QOS='yes'
QOS_INTERNET_DEV_N='1'
QOS_INTERNET_DEV_1='ppp0'
QOS_INTERNET_BAND_DOWN='768Kibit/s'
QOS_INTERNET_BAND_UP='128Kibit/s'
QOS_INTERNET_DEFAULT_DOWN='4'
QOS_INTERNET_DEFAULT_UP='0'

QOS_CLASS_N='4'

QOS_CLASS_1_PARENT='0'
QOS_CLASS_1_MINBANDWIDTH='232Kibit/s'
QOS_CLASS_1_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_1_DIRECTION='down'
QOS_CLASS_1_PRIO=''

QOS_CLASS_2_PARENT='0'
QOS_CLASS_2_MINBANDWIDTH='232Kibit/s'
QOS_CLASS_2_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_2_DIRECTION='down'
QOS_CLASS_2_PRIO=''

QOS_CLASS_3_PARENT='0'
QOS_CLASS_3_MINBANDWIDTH='232Kibit/s'
QOS_CLASS_3_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_3_DIRECTION='down'
QOS_CLASS_3_PRIO=''

QOS_CLASS_4_PARENT='0'
QOS_CLASS_4_MINBANDWIDTH='72Kibit/s'
QOS_CLASS_4_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_4_DIRECTION='down'
QOS_CLASS_4_PRIO=''


QOS_FILTER_N='3'

QOS_FILTER_1_CLASS='1'
QOS_FILTER_1_IP_INTERN='192.168.0.2'
QOS_FILTER_1_IP_EXTERN=''
QOS_FILTER_1_PORT=''
QOS_FILTER_1_PORT_TYPE=''
```

```
QOS_FILTER_1_OPTION=''

QOS_FILTER_2_CLASS='2'
QOS_FILTER_2_IP_INTERN='192.168.0.3'
QOS_FILTER_2_IP_EXTERN=''
QOS_FILTER_2_PORT=''
QOS_FILTER_2_PORT_TYPE=''
QOS_FILTER_2_OPTION=''

QOS_FILTER_3_CLASS='3'
QOS_FILTER_3_IP_INTERN='192.168.0.4'
QOS_FILTER_3_IP_EXTERN=''
QOS_FILTER_3_PORT=''
QOS_FILTER_3_PORT_TYPE=''
QOS_FILTER_3_OPTION=''
```

Option `QOS_INTERNET_DEFAULT_UP` is set to 0 because upstream should not be regulated.

Example 2

This example targets at spreading bandwidth between 2 client PCs and then those client bandwidths between one port and the remaining traffic on the client.

We create 2 classes at first with the complete speed for each client and attach them directly to the interface for "up" res. "down" (see example 1). No we create two additional classes for the first client attached to the first class. These classes are created in the same way like the first ones directly attached to the interface except for one difference: `QOS_CLASS_x_PARENT` is not 0 but the number of the parent class it is attached to. If this for example is `QOS_CLASS_1` the classes' `QOS_CLASS_1` has to be set to 1. The same is applied for the second client PC. Attach two subclasses to the class for the second PC. This could be done for an infinite number of PCs if needed. Subclasses of a class can also be created in the amount needed.

This is our skeleton. Now filters have to be defined to assign traffic to the classes (see example 1).

2 filters have to be created for each client. One filter on a port and one for the remaining traffic of the client. Filter sequence is of essential importance here. First filter the port and then the rest. The other way round the filter for the remaining traffic would assign all traffic to its class (including the port traffic).

Let's call the interface 0, the 6 classes 1, 2, 3, 4, 5, and 6 and the 4 filters F1, F2, F3 and F4. The scenario looks like this image 4.7.

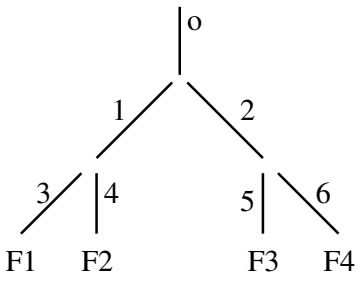


Figure 4.8.: Example 2

Configuration looks like this:

2 classes for 2 PCs getting 1/2 interface bandwidth each with 2 classes for a port getting 2/3 and the rest getting 1/3 of its parent class:

```
OPT_QOS='yes'
QOS_INTERNET_DEV_N='1'
QOS_INTERNET_DEV_1='ppp0'
QOS_INTERNET_BAND_DOWN='768Kibit/s'
QOS_INTERNET_BAND_UP='128Kibit/s'
QOS_INTERNET_DEFAULT_DOWN='7'
QOS_INTERNET_DEFAULT_UP='0'

QOS_CLASS_N='6'

QOS_CLASS_1_PARENT='0'
QOS_CLASS_1_MINBANDWIDTH='384Kibit/s'
QOS_CLASS_1_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_1_DIRECTION='down'
QOS_CLASS_1_PRIO=''

QOS_CLASS_2_PARENT='0'
QOS_CLASS_2_MINBANDWIDTH='384Kibit/s'
QOS_CLASS_2_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_2_DIRECTION='down'
QOS_CLASS_2_PRIO=''

QOS_CLASS_3_PARENT='1'
QOS_CLASS_3_MINBANDWIDTH='256Kibit/s'
QOS_CLASS_3_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_3_DIRECTION='down'
QOS_CLASS_3_PRIO=''

QOS_CLASS_4_PARENT='1'
QOS_CLASS_4_MINBANDWIDTH='128Kibit/s'
QOS_CLASS_4_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_4_DIRECTION='down'
QOS_CLASS_4_PRIO=''

QOS_CLASS_5_PARENT='2'
QOS_CLASS_5_MINBANDWIDTH='256Kibit/s'
QOS_CLASS_5_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_5_DIRECTION='down'
QOS_CLASS_5_PRIO=''

QOS_CLASS_6_PARENT='2'
QOS_CLASS_6_MINBANDWIDTH='128Kibit/s'
QOS_CLASS_6_MAXBANDWIDTH='768Kibit/s'
QOS_CLASS_6_DIRECTION='down'
QOS_CLASS_6_PRIO=''

QOS_FILTER_N='4'

QOS_FILTER_1_CLASS='3'
```

```
QOS_FILTER_1_IP_INTERN='192.168.0.2'
QOS_FILTER_1_IP_EXTERN=''
QOS_FILTER_1_PORT='80'
QOS_FILTER_1_PORT_TYPE='client'
QOS_FILTER_1_OPTION=''

QOS_FILTER_2_CLASS='4'
QOS_FILTER_2_IP_INTERN='192.168.0.2'
QOS_FILTER_2_IP_EXTERN=''
QOS_FILTER_2_PORT=''
QOS_FILTER_2_PORT_TYPE=''
QOS_FILTER_2_OPTION=''

QOS_FILTER_3_CLASS='5'
QOS_FILTER_3_IP_INTERN='192.168.0.3'
QOS_FILTER_3_IP_EXTERN=''
QOS_FILTER_3_PORT='80'
QOS_FILTER_3_PORT_TYPE='client'
QOS_FILTER_3_OPTION=''

QOS_FILTER_4_CLASS='6'
QOS_FILTER_4_IP_INTERN='192.168.0.3'
QOS_FILTER_4_IP_EXTERN=''
QOS_FILTER_4_PORT=''
QOS_FILTER_4_PORT_TYPE=''
QOS_FILTER_4_OPTION=''
```

Option `QOS_INTERNET_DEFAULT_DOWN` was set in a way that traffic not being assigned to a class by a filter is put in a non-existent class. This is to simplify the example and because it is assumed that there is no unassigned traffic left. Traffic being sent to a non-existent class is forwarded very slow. If a rest of traffic exists always ensure that it is assigned to its own (existing) class.

Option `QOS_INTERNET_DEFAULT_UP` was set to 0 because upstream should not be regulated.

Example 3

An example targeting at learning the general structures of working with QoS.

Picture 4.9 once again shows the layout of example two but this time with an extension. Both subclasses of the second class have two more subclasses attached. You see that it is possible to have nested subclasses. The maximum for nested subclasses is a depth of 8 where 0 is the interface itself leaving 7 more possible subclass levels. "Width" is unlimited though. A subclass can have an unlimited number of classes attached.

The picture shows as well that it is possible to have more than one filter attached to a class as it is done in class 10. Pay attention that at the moment it is not possible to attach a filter in the middle of the "tree" as F8 intended to.

Lets have a closer look at the sense of classes and subclasses. Classes set and control speed of a connection. Spreading of speed is handled described as in `QOS_CLASS_x_MINBANDWIDTH`. This can have disadvantages if i.e. you attach all classes to one parent class. If it is for example intended that one client PC should have half of the available bandwidth and the other half

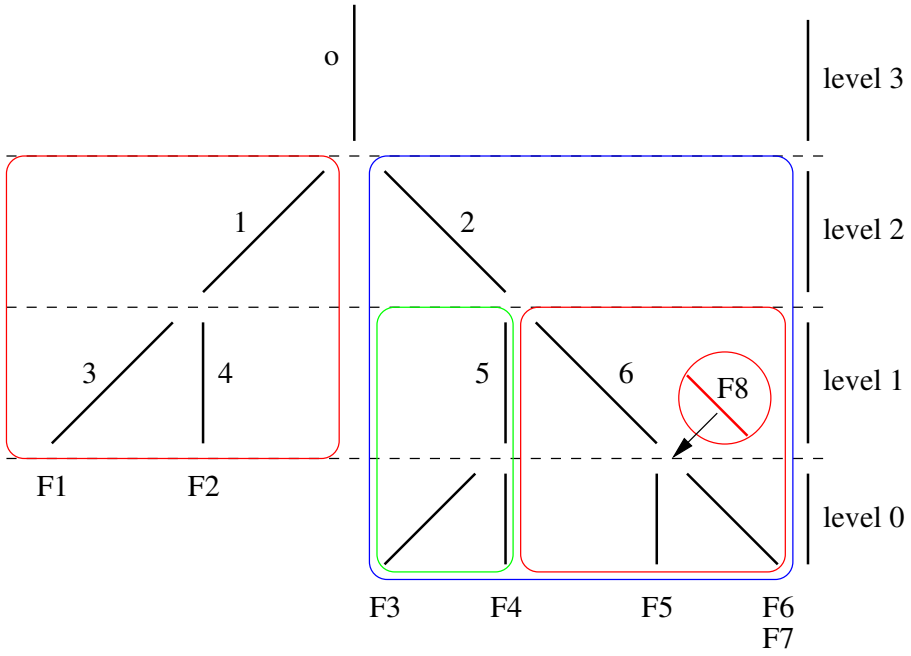


Figure 4.9.: Example 3

is for a second client PC divided in 2/3 http and 1/3 for the rest (2/6 and 1/6 of the whole bandwidth) this would happen: If both clients run at full load both get their half of the bandwidth. If the second one is not transferring http 2/6 of unused bandwidth are distributed not only to the second but to both PCs as decribed above. To avoid this subclasses are created. Traffic of a class is at first distributed to its subclasses. Only if they don't use the complete traffic the rest is spread to the other classes. In the picture the areas that belong together are encircled (red = 1, blue = 2, green = 5 and orange = 6.

Example 4

Configuration for ACK packet priorization in order to keep downstream high if upstream has heavy load:

```
OPT_QOS='yes'
QOS_INTERNET_DEV_N='1'
QOS_INTERNET_DEV_1='ppp0'
QOS_INTERNET_BAND_DOWN='768Kibit/s'
QOS_INTERNET_BAND_UP='128Kibit/s'
QOS_INTERNET_DEFAULT_DOWN='0'
QOS_INTERNET_DEFAULT_UP='2'
```

Configure ppp0 as the Internet device (DSL) and give it the usual up/downstream bandwidth for TDSL (and some other providers). It may be necessary to lower upstream bandwidth for some Kibibit for the trying.

No classes for downstream should be defined:

```
QOS_INTERNET_DEFAULT_DOWN='0'
```

For upstream class number two should be the default class. The network device eth0 is set to 10Mibit/s.

```
   QOS_CLASS_N='2'

QOS_CLASS_1_PARENT='0'
QOS_CLASS_1_MINBANDWIDTH='127Kibit/s'
QOS_CLASS_1_MAXBANDWIDTH='128Kibit/s'
QOS_CLASS_1_DIRECTION='up'
QOS_CLASS_1_PRIO=''
```

This is the class for ACK (acknowledgement) packets. ACK packets are rather small and thus need only a minimum bandwidth. Because they should not be affected in any way they get 127Kibit/s. 1Kibit/s is left for the rest.

```
QOS_CLASS_2_PARENT='0'
QOS_CLASS_2_MINBANDWIDTH='1Kibit/s'
QOS_CLASS_2_MAXBANDWIDTH='128Kibit/s'
QOS_CLASS_2_DIRECTION='up'
QOS_CLASS_2_PRIO=''
```

This class is for everything else (except ACK packets). The bandwidth that is left is 1Kibit/s (128-127=1). We don't limit it to 1Kibit/s though, the class is limited by the entry

```
QOS_CLASS_2_MAXBANDWIDTH='128Kibit/s'
```

Because our first class never can use all of its bandwidth there will be something left over which then gets allocated to the second class. If upstream should be divided some more (prominent use case) all other classes have to be subclasses "under" this class. Of course `QOS_INTERNET_DEFAULT_UP` has to be adapted then.

```
QOS_FILTER_N='1'

QOS_FILTER_1_CLASS='1'
QOS_FILTER_1_IP_INTERN=''
QOS_FILTER_1_IP_EXTERN=''
QOS_FILTER_1_PORT=''
QOS_FILTER_1_PORT_TYPE=''
QOS_FILTER_1_OPTION='ACK'
```

This filter filters all packets matching option ACK (i.e. ACK packets). By specifying `QOS_FILTER_1_CLASS`='1' we achieve that all these packets filtered are sent to class 1.

For testing purposes look for one or more good up- and download sources that can produce full load for up- as well as for downstream. Have a look at the traffic display in ImonC. Try this once with and once without QoS.

Downstream should not or not as much decline as without this configuration. It could get even better by declining upstream bandwidth in steps of Kibibits and analyze the effect. I reached my optimum at 121Kibit/s (no declining downstreams anymore). Of course MAXBANDWIDTH- and MINBANDWIDTH- values of all classes have to be adapted accordingly.

4.19. SSHD - Secure Shell, Secure Copy

A secure shell enables you to open an encrypted connection with the fli4l router. By using secure copy files can be transmitted encrypted to the fli4l router. If in addition Public Key Login (Page 219) is used commands and file transfers can be executed driven by scripts from "outside". As of version 2.1.7 only a SSH2 server is existing.

4.19.1. Installation Of The Secure-Shell-Daemon

OPT_SSHD Default setting: `OPT_SSHD='no'`

If the router should be accessible via ssh set `OPT_SSHD` to `'yes'`. This will install the ssh server Dropbear on the fli4l router. It will also enable copying of files to the router.

SSHD_ALLOWPASSWORDLOGIN Default setting: `SSHD_ALLOWPASSWORDLOGIN='yes'`

If `SSHD_ALLOWPASSWORDLOGIN` is set to `'no'` fli4l won't allow ssh login via password anymore. Login can only be done via private/public key. This assumes that a public key (Page 219) is present on the router.

SSHD_CREATEHOSTKEYS Default setting: `SSHD_CREATEHOSTKEYS='no'`

A ssh server needs a so-called host key that is unique to identify itself to a ssh client. The package SSHD provides a host key to allow a first login to the router but this key should be replaced with a self-generated one only known to you as fast as possible. Generating your own host key is the only way to be prepared against so called man-in-the-middle attacks and thus is very important. SSH will notice if someone pretends to be your fli4l router because his host key will differ and will warn you about the host key changing.

Generating your own host key will be done automatically if `SSHD_CREATEHOSTKEYS` is set to `'yes'`. This is a challenging task and can prolong boot time for several minutes. If the fli4l router starts with `SSHD_CREATEHOSTKEYS` activated one (or more) host key(s) will be created in the directory `/tmp/ssh`. Keyfiles found there have to be copied over to your fli4l build directory under `etc/ssh` (on the PC where fli4l's boot medium is created). In my case a directory listing of config.babel looks like this:

Please note that under the directory `config.babel` a subdirectory etc exists with another subdirectory ssh. Generated host keys have to be placed there. As of fli4l version 2.1.5 files in your config directory will be preferred over the ones from the opt directory. With the next update of your fli4l boot medium the files from `config/etc/ssh` will be integrated and not those in `opt/etc/ssh`. In this way every fli4l router you configure can have its own unique host key. When creating the fli4l files there will appear a message „appending config specific files to opt.img ..." towards the end. All files coming from the config directory instead of opt will be listed there.

```
#
# appending config specific files to opt.img ...
#
etc/ssh/dropbear_dss_host_key
etc/ssh/dropbear_rsa_host_key
```

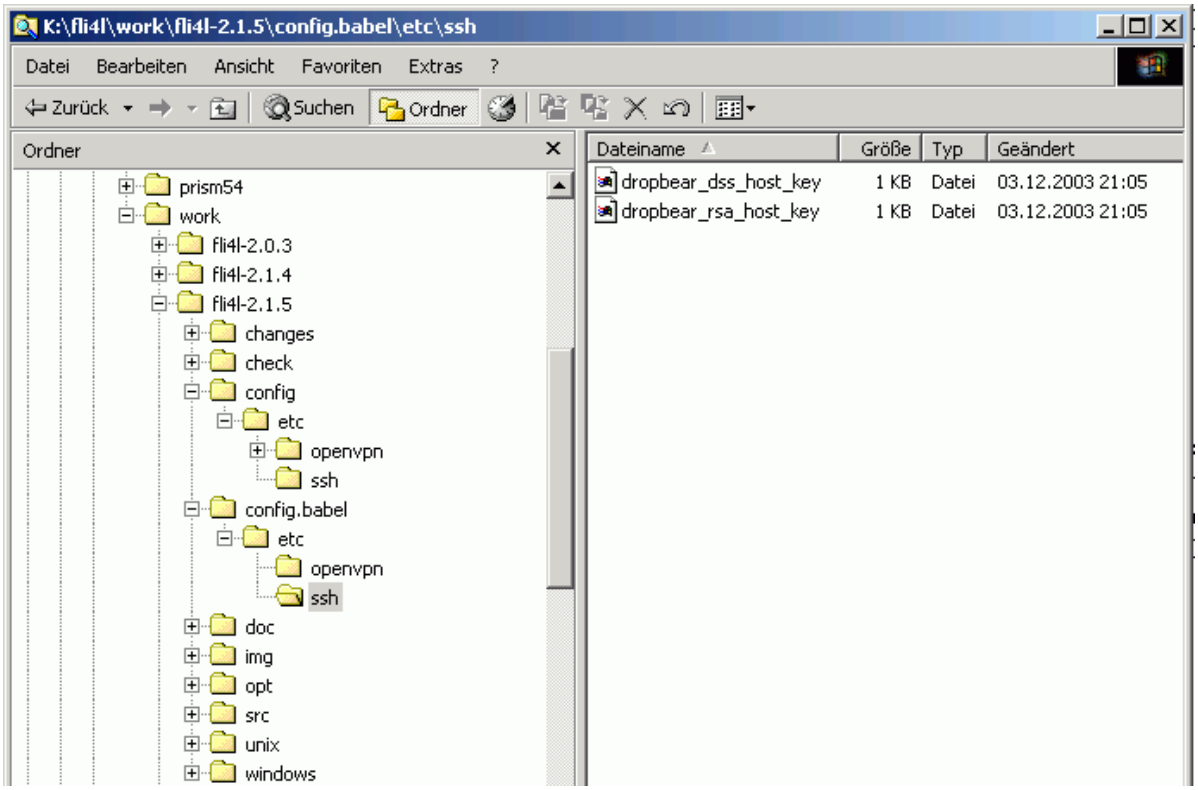


Figure 4.10.: Directory structure of fli4l

If you created a new host key set SSHD_CREATEHOSTKEYS back to 'no' to avoid creating another host key on every reboot.

If you log in to your fli4l router after updating the host key a warning message (depending on the ssh client you use) will appear to inform you about the changed host key. In this case this is normal because you just changed your host key. Follow the routine necessary for your ssh client to accept the changed host key permanently. If some time in the future you see this warning again you will have to check why it appears. Don't just accept a changed host key blindly!

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
ca:a4:ab:e7:af:d8:68:05:d3:1f:e6:15:08:d6:ed:36.
Please contact your system administrator.
Add correct host key in /home/babel/.ssh/known_hosts to get rid of this message.
Offending key in /home/babel/.ssh/known_hosts:7
Password authentication is disabled to avoid man-in-the-middle attacks.
```

SSHD_PORT Default setting: SSHD_PORT='22'

By SSHD_PORT a non-standard port can be defined the ssh server should listen to.

If ssh login from outside should be allowed INPUT_ACCEPT_PORT_x (Page 40) has to be adapted to reflect the change.

The commands accessing fli4l from an Unix-/Linux client over protocol SSH are:

- ssh - Secure Shell
- scp - Secure Copy

Corresponding programs are available for Windows as well, see:
http://www.chiark.greenend.org.uk/~sgtatham/putty/
http://winscp.net/eng/docs/lang:en
http://www.tectia.com/de/de.iw3

SSHD_PUBLIC_KEY_N Default setting: SSHD_PUBLIC_KEY_N='0'

SSHD_PUBLIC_KEY_N holds the number of public keys to be copied to the fli4l router.

SSH allows authentification based on asymmetric encryption. Authentification is done via username and public/private key instead of username and password. This way entering a password can be omitted. Generate your key pair by the help of ssh-keygen (or puttygen if putty under Windows is used as the ssh client). When generating keys you can optionally specify a passphrase (a password for using the key) to increase security even more. If using a passphrase you may consider working with an ssh agent (ssh-agent or pageant).

Important: *The private part of the keypair has to be guarded as careful as a password because it has the same function. The private part of your keypair is only known to your*

ssh client. The public part of the key will be needed on the fli4l router and is provided to it by SSHD_PUBLIC_KEY_x or SSHD_PUBLIC_KEYFILE_x.

For further informations see manual pages for ssh and its components res. the documentation for putty (http://www.chiark.greenend.org.uk/~sgtatham/putty/).

SSHD_PUBLIC_KEY_x Provide the public part of each user's key here who should be able to access flil4 via ssh. The easiest way is cut and paste it from a terminal window. Example:

```
SSHD_PUBLIC_KEY_1='1024 ... username@hostname'
```

Important: *The key does not contain carriage returns. Puttygen will insert those eventually while doing cut-and-paste. They will have to be deleted again.*

SSHD_PUBLIC_KEYFILE_N Default setting: `SSHD_PUBLIC_KEYFILE_N='0'`

Instead of copying the content of the public part of the key to sshd.txt you could copy it directly to the opt archive. This works like described for `SSH_CREATEHOSTKEYS`. Copy the public part of the key to the directory `<config>/etc/ssh`.

SSHD_PUBLIC_KEYFILE_x The file name of the public key in directory `<config>/etc/ssh`.

```
SSHD_PUBLIC_KEYFILE_1='root@fli4l'
```

SSH_CLIENT_PRIVATE_KEYFILE_N Default setting: `SSH_CLIENT_PRIVATE_KEYFILE_N='0'`

If you want to use private keys for the ssh or plink client for login at a ssh server you could copy them to the directory `<config>/etc/ssh`. This works the same way as described in `SSH_CREATEHOSTKEYS`. Copy your private key to the directory `<config>/etc/ssh`. Private keys in OpenSSH format will be automatically converted to dropbear format on each boot.

SSH_CLIENT_PRIVATE_KEYFILE_x The file name of the private key in directory `<config>/etc/ssh`.

```
SSHD_PRIVATE_KEYFILE_1='babel@rootserver'
```

4.19.2. Installation Of Dbclient

OPT_SSH_CLIENT Default setting: `OPT_SSH_CLIENT='no'`

To use a pure ssh2/scp client activate dbclient from dropbear by setting `OPT_SSH_CLIENT` to `'yes'`. The advantage of this client is that it shares program code with the dropbear ssh server. This saves a lot of space in the OPT archive. Dbclient is more or less compatible with ssh/scp client, its command syntax is similar. A symbolic link to /usr/bin/ssh res. /usr/bin/scp will be created to make ssh <host> res. scp <source> <target> working out of the box.

If dbclient's known hosts should be saved permanently the file known_hosts from the directory /.ssh on the router has to be copied to config/etc/ssh. This works in the same as with a generated host key. In the following example the fli4l directory (fli4l's boot medium is generated there) is found at /home/babel/fli4l-3.10.5 . All config files are in directory config.babel.

```
cd /home/babel/fli4l-3.10.5
mkdir -p config.babel/etc/ssh
scp fli4l:/.ssh/* config.babel/etc/ssh
```

4.19.3. Installation Of A Plink Client

OPT_PLINK_CLIENT Default setting: `OPT_PLINK_CLIENT='no'`

Installs a ssh1/ssh2/telnet client on the fli4l router. plink is the Unix version of the well known PuTTY program for Windows. Executing plink on the fli4l router displays a help page for using plink.

If plink's known hosts should be saved permanently the file sshhostkeys from the directory /.putty on the router has to be copied to <config>/etc/plink. This works in the same as with a generated host key. In the following example the fli4l directory (fli4l's boot medium is generated there) is found at /home/babel/fli4l-3.10.5 . All config files are in directory config.babel.

```
cd /home/babel/fli4l-3.10.5
mkdir -p config.babel/etc/plink
scp fli4l:/.putty/* config.babel/etc/plink
```

4.19.4. Installation Of A Sftp Server

OPT_SFTPSERVER Default setting: `OPT_SFTPSERVER='no'`

Installs a sftp server on the fli4l router.

4.19.5. Literature

Dropbear SSH2 Site: http://matt.ucc.asn.au/dropbear/dropbear.html

4.20. TOOLS - Additional Tools For Debugging

Package TOOLS provides some Unix programs mostly used for administration and debugging. Other ones like wget are used i.e. for catching the first (ad-)page of some providers. By setting the value 'yes' the mentioned program is copied to the fli4l router. Default setting is 'no'. The programs are only described in short, how to use them is described in their repective man pages which can be found in your favourite linux distribution or online at: http://www.linuxmanpages.com

4.20.1. Networking-Tools

OPT_DIG DNS Multitool

The command dig allows to execute various DNS queries.

OPT_FTP FTP-Client

The ftp program can connect fli4l to a FTP server to move files between the two of them.

FTP_PF_ENABLE_ACTIVE The setting `FTP_PF_ENABLE_ACTIVE='yes'` adds a rule to the packet filter that enables initiated active FTP. For `FTP_PF_ENABLE_ACTIVE='no'` such a rule has to be added to the `PF_OUTPUT_%`-array manually (if needed), for an example look here (Page 64).

Passive FTP is always possible, neither this variable nor an explicit packet filter rule is needed then.

OPT_IFTOP Netzwerküberwachung

The iftop program lists all active network connections and their throughput directly on the fli4l router.

After login to the router iftop is simply started by executing it on the console.

OPT_IMONC Text based control program for imond

This program is a text based frontend for the router to control imond.

OPT_IPERF Performance checks for the network

The command iperf can do performance measuring for networks. The program has to be started on the two machines involved. On the iperf server execute

```
fli4l-server 3.10.5~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

to start. The server will wait for a connection from an iperf client then. Start the client on the second machine with

```
fli4l-client 3.10.5~# iperf -c 1.2.3.4
------------------------------------------------------------
Client connecting to 1.2.3.4, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 1.2.3.5 port 50311 connected with 1.2.3.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec    985 MBytes    826 Mbits/sec
```

A performance check will start immedeately showing first results. Some options can be added to iperf, for details evaluate the informations on its homepage http://iperf.sourceforge.net/.

OPT_NETCAT Transfer data to TCP based servers

OPT_NGREP A grep that is able to work directly with network devices.

OPT_NTTCP Network checks

The program NTTCP can check network speed. On one side a server is started and on the other side the client.

Start the server by executing nttcp -i -v. The server will wait for client requests. To test i.e. speed execute nttcp -t <IP Adress of the server> on the client.

This is how a started nttcp server looks like:

```
fli4l-server 3.10.5~# nttcp -i -v
nttcp-l: nttcp, version 1.47
nttcp-l: running in inetd mode on port 5037 - ignoring options beside -v and -p
```

This is how a test with a nttcp client looks like:

```
fli4l-client 3.10.5~# nttcp -t 192.168.77.77
l~~8388608~~~~4.77~~~~0.06~~~~~14.0713~~~1118.4811~~~~2048~~~~429.42~~~34133.3
1~~8388608~~~~4.81~~~~0.28~~~~~13.9417~~~~239.6745~~~~6971~~~1448.21~~~24896.4
```

The nttcp help shows all further parameters:

```
Usage: nttcp [local options] host [remote options]
       local/remote options are:
        -t      transmit data (default for local side)
        -r      receive data
        -l#     length of bufs written to network (default 4k)
        -m      use IP/multicasting for transmit (enforces -t -u)
        -n#     number of source bufs written to network (default 2048)
        -u      use UDP instead of TCP
        -g#us   gap in micro seconds between UDP packets (default 0s)
        -d      set SO_DEBUG in sockopt
        -D      don't buffer TCP writes (sets TCP_NODELAY socket option)
        -w#     set the send buffer space to #kilobytes, which is
                dependent on the system - default is 16k
        -T      print title line (default no)
        -f      give own format of what and how to print
        -c      compares each received buffer with expected value
        -s      force stream pattern for UDP transmission
        -S      give another initialisation for pattern generator
        -p#     specify another service port
        -i      behave as if started via inetd
        -R#     calculate the getpid()/s rate from # getpid() calls
        -v      more verbose output
        -V      print version number and exit
        -?      print this help
        -N      remote number (internal use only)
        default format is: %9b%8.2rt%8.2ct%12.4rbr%12.4cbr%8c%10.2rcr%10.1ccr
```

OPT_RTMON Installs a tool that will track changes in routing tables. Primary used for debugging.

OPT_SOCAT The program "socat" is more or less an enhanced version of the "netcat" program (Page 222) with more functionality. By using "socat" you may not only establish or accept various types of network connections, but also sent data to or read data from UNIX sockets, devices, FIFOs, and so on. In addition sources and destinations of *different* types may be connected: An example would be a Network server listening on a TCP port and then writing received data to a local FIFO or reading data from the FIFO and then transmitting it over the network to a client. See http://www.dest-unreach.org/socat/doc/socat.html for more information and application examples.

OPT_TCPDUMP debug

The program tcpdump can watch, interpret and record network traffic. Find more on this by feeding Google with "tcpdump man"

tcpdump <parameter>

OPT_DHCPDUMP DHCP packet dumper

The program "dhcpdump" can be used to analyse DHCP packets more accurately. It is based on tcpdump and produces easily readable output.

Usage:

```
dhcpdump -i interface [-h regular-expression]
```

The program can be executed using the following command:

```
dhcpdump -i eth0
```

If desired, the analysis can be cut down to a specific MAC address using regular expressions. The command would be:

```
dhcpdump -i eth0 -h ^00:a1:c4
```

The output could look like this:

```
  TIME: 15:45:02.084272
    IP: 0.0.0.0.68 (0:c0:4f:82:ac:7f) > 255.255.255.255.67 (ff:ff:ff:ff:ff:ff)
    OP: 1 (BOOTPREQUEST)
 HTYPE: 1 (Ethernet)
  HLEN: 6
  HOPS: 0
   XID: 28f61b03
  SECS: 0
 FLAGS: 0
CIADDR: 0.0.0.0
YIADDR: 0.0.0.0
```

```
SIADDR: 0.0.0.0
GIADDR: 0.0.0.0
CHADDR: 00:c0:4f:82:ac:7f:00:00:00:00:00:00:00:00:00:00
 SNAME: .
 FNAME: .
OPTION:  53 (  1) DHCP message type         3 (DHCPREQUEST)
OPTION:  54 (  4) Server identifier         130.139.64.101
OPTION:  50 (  4) Request IP address        130.139.64.143
OPTION:  55 (  7) Parameter Request List      1 (Subnet mask)
                                              3 (Routers)
                                             58 (T1)
                                             59 (T2)
```

OPT_WGET http/ftp Client

The program wget can fetch data from web servers in batch mode. More useful is (and that is why wget is included here) that it can catch redirections to your provider's own webserver while establishig a connection to the internet in a simple way i.e. for Freenet. A (german) Mini-HowTo on this topic can be found here:
http://www.fli4l.de/hilfe/howtos/einsteiger/wget-und-freenet/

4.20.2. Hardware Identification

Often you do not know exactly what hardware is in your own computer or which driver you should use for i.e. your network card or the USB chipset. The hardware itself can help here. It provides a list of devices in the computer and the associated driver if possible. You can choose whether the recognition is done at boot (which is recommended before the first installation) or later when the computer is running, comfortably triggered from the Web interface. The output might look like this:

```
fli4l 3.10.5 # cat /bootmsg.txt
#
# PCI Devices and drivers
#
Host bridge: Advanced Micro Devices [AMD] CS5536 [Geode companion] Host Bridge (rev 33)
Driver: 'unknown'
Entertainment encryption device: Advanced Micro Devices [AMD] Geode LX AES Security Block
Driver: 'geode_rng'
Ethernet controller: VIA Technologies, Inc. VT6105M [Rhine-III] (rev 96)
Driver: 'via_rhine'
Ethernet controller: VIA Technologies, Inc. VT6105M [Rhine-III] (rev 96)
Driver: 'via_rhine'
Ethernet controller: VIA Technologies, Inc. VT6105M [Rhine-III] (rev 96)
Driver: 'via_rhine'
Ethernet controller: Atheros Communications, Inc. AR5413 802.11abg NIC (rev 01)
Driver: 'unknown'
ISA bridge: Advanced Micro Devices [AMD] CS5536 [Geode companion] ISA (rev 03)
Driver: 'unknown'
IDE interface: Advanced Micro Devices [AMD] CS5536 [Geode companion] IDE (rev 01)
Driver: 'amd74xx'
USB Controller: Advanced Micro Devices [AMD] CS5536 [Geode companion] OHC (rev 02)
```

```
Driver: 'ohci_hcd'
USB Controller: Advanced Micro Devices [AMD] CS5536 [Geode companion] EHC (rev 02)
Driver: 'ehci_hcd'
```

In essential 3 network cards are in this machine, driven by 'via_rhine' and an Atheros-WiFi card driven by madwifi (the driver name is not resoved correctly here).

OPT_HW_DETECT This variable triggers that the files needed for hardware identification are copied to the router. Results can either be found on the console while booting if HW_DETECT_AT_BOOTTIME is set to 'yes' or in the web interface if OPT_HTTPD (Page 121) was set to 'yes'. The content of '/bootmsg.txt' can of course be checked in the web interface as well if you have a working network connection to the router.

HW_DETECT_AT_BOOTTIME Starts hardware identification start on boot. Recognition will take place in background (it will take some time) and then writes its output to console and '/bootmsg.txt'.

OPT_LSPCI Listing of all PCI devices

OPT_I2CTOOLS Tools for I^2C access.

OPT_IWLEEPROM Tool to access the EEPROM of Intel and Atheros WLAN cards.

Needed e.g. to reprogram the regulatory domain on ath9k cards (see http://blog.asiantuntijakaveri.fi/2014/08/one-of-my-atheros-ar9280-minipcie-cards.html).

OPT_ATH_INFO Hardware diagnosis tool for WLAN cards based on Atheros chipset.

This tool can extract detailed information about the hardware used in Atheros wireless cards, e.g. ath5k. These include, amongst others, the chipset used or calibration information.

4.20.3. File Management Tools

OPT_E3 An editor for fli4l

E3 is a very small editor written in Assembler. It mimics various editor modes known from other („full size") editors. To choose a mode e3 only has to be started with the right command. A short key overview is given by e3 starting without parameter or by pressing Alt+H (except in VI-mode, press „:h" in CMD mode then). Caret (^) stands for the Ctrl-/Strg key.

Command	Mode
e3 / e3ws	WordStar, JOE
e3vi	VI, VIM
e3em	Emacs
e3pi	Pico
e3ne	NEdit

OPT_MTOOLS mtools provide some DOS-like commands for simpler handling of DOS media (copying, formatting, a.s.o.).

Exact syntax of the commands can be found in the mtools documentation:
http://www.gnu.org/software/mtools/manual/mtools.html

OPT_SHRED Installs the program *shred* on the router which is used for secure erasing of block devices.

OPT_YTREE File Manager

Installs the File Manager Ytree on the router.

4.20.4. Developer-Tools

OPT_OPENSSL The tool openssl can i.e. be used to test crypto accelerator devices.

```
openssl speed -evp des -elapsed
openssl speed -evp des3 -elapsed
openssl speed -evp aes128 -elapsed
```

OPT_STRACE debug

By using the program strace you can trace system calls and signals or watch function calls and runtime behavior of a program.

strace <program>

OPT_REAVER Brute force attacks on Wifi WPS PINs

Tests all possible WPS PINS to find the WPA password. For details on usage see: http://code.google.com/p/reaver-wps/.

OPT_VALGRIND Installs valgrind on the router.

4.21. UMTS - Internet Connection Via UMTS

Connecting a fli4l to the Internet via UMTS. For proper operation other packages may be needed.

4.21.1. Configuration

OPT_UMTS Default setting: `OPT_UMTS`='no'

'yes' activates the package.

UMTS_DEBUG Default setting: `UMTS_DEBUG`='no'

If pppd should output additional debug information set `UMTS_DEBUG` to 'yes'. In this case pppd will write additional informations to the syslog interface.

IMPORTANT: To see this messages via syslogd `OPT_SYSLOGD` has to be set to 'yes'.

UMTS_PIN Default setting: `UMTS_PIN`='disabled'

Pin for the SIM card

Give a four digit number or the word 'disabled'

UMTS_DIALOUT Default setting: `UMTS_DIALOUT`='*99***1#'

Dialing parameters for the connection

Dial-in data of some german providers

Provider	APN	Username	Password
T-Mobile	internet.t-mobile	arbitrary	arbitrary
Vodafone	web.vodafone.de	arbitrary	arbitrary
E-Plus	internet.eplus.de	eplus	gprs
O2 (Vertragskunden)	internet	arbitrary	arbitrary
O2 (Prepaid-Kunden)	pinternet.interkom.de	arbitrary	arbitrary
Alice	internet.partner1	arbitrary	arbitrary

UMTS_GPRS_UMTS Default setting: `UMTS_GPRS_UMTS`='both'

Which transfer mode should be used

Possible values (both, gprs, umts)

UMTS_APN Default setting: `UMTS_APN`='web.vodafone.de'

UMTS_USER Default setting: `UMTS_USER`='anonymer'

UMTS_PASSWD Default setting: `UMTS_PASSWD`='surfer'

Specify data needed for dial-in.

Set username and password for the provider used. `UMTS_USER` is the username, `UMTS_PASSWD` the password.

Name of the APN (Acess Provider Node) for some german providers

- http://www.teltarif.de/mobilfunk/internet/einrichtung.html

UMTS_NAME Default setting: `UMTS_NAME`='UMTS'

Set a name for the circuit here - maximum 15 chars. It will be shown in the imon-client imonc. Blanks are not allowed.

UMTS_HUP_TIMEOUT Default setting: `UMTS_TIMEOUT`='600'

Specify a hangup time in seconds here if no traffic is detected over the UMTS connection. A timeout '0' is equal to no timeout.

UMTS_TIMES Default setting: `UMTS_TIMES`='Mo-Su:00-24:0.0:Y'

Times mentioned here determine when the circuit becomes active and at what costs. This allows to use different circuits and default routes at different times (Least-Cost-Routing). The daemon imond will control routing.

UMTS_CHARGEINT Default setting: `UMTS_CHARGEINT`='60'

Charge-Interval: Timespan in seconds, that will be used for calculating online costs.

UMTS_USEPEERDNS Default setting: `UMTS_USEPEERDNS`='yes'

Use the provider's DNS server or not.

UMTS_FILTER Default setting: `UMTS_FILTER='yes'`

fli4l automatically hangs up if no traffic is going over the ppp0 interface in the hangup timeout time. Unfortunately also data transfers from outside count as relevant traffic i.e. P2P-clients like eDonkey. Since you will be nearly permanently contacted from outside nowadays it may happen that fli4l never hangs up an UMTS connection.

Option `UMTS_FILTER` is helping here. If set to 'yes' only traffic generated by your own machine is monitored and external traffic will be ignored completely. Since incoming traffic normally leads to a reaction from the router or the machines behind it (i.e. denying or dropping the packets) some additional outgoing packets will be ignored too.

UMTS_ADAPTER (optional)

Specify here if the hardware is a PCMCIA card, an USB adapter or a phone connected via an USB cable.

If omitting this variable only the files necessary for an USB adapter will be copied.

Possible values: (pcmcia,usbstick,usbphone)

All variables that follow are optional and only needed if the automatic detection fails.

UMTS_IDVENDOR (optional) `UMTS_IDVENDOR='xxxx'`

Vendor ID after plugging/starting the adapter

UMTS_IDDEVICE (optional) `UMTS_IDDEVICE='xxxx'`

Product ID after plugging/starting the adapter

Specifying the following two parameters is only needed if the ID changes after initialisation

UMTS_IDVENDOR2 (optional) `UMTS_IDVENDOR2='xxxx'`

Vendor ID after initialisation of the adapter

UMTS_IDDEVICE2 (optional) `UMTS_IDDEVICE2='xxxx'`

Product ID after initialisation of the adapter

UMTS_DRV (optional) `UMTS_DRV='xxxx'`

Driver for the adapter, if omitted 'usbserial' is used

UMTS_SWITCH (optional) `UMTS_SWITCH='-v 0x0af0 -p 0x6971 -M 555...000 -s 10'`

Parameters for usb-modeswitch initialisation of the modem (see Website usb-modeswitch). With a few exceptions all modems mentioned on the website should be recognized automatically.

- http://www.draisberghof.de/usb_modeswitch/

UMTS_DEV (optional)

In case of problems the data interface for pppd can be set here. The most usual for adapters are:

```
ttyUSB0 for usbstick
ttyS2   for pcmcia
ttyACM0 for usbphone
```

UMTS_CTRL (optional)

Some adapter have more interfaces for modem control. If only one is existig status informations can only be read in 'Offline' state. For Option Fusion UMTS Quad the interface is i.e. ttyUSB2.

4.22. USB - Support For USB Devices

OPT_USB Supprt for USB devices in general is activated here. Only if 'yes' is entered here USB devices will get usable at all. If you have chosen an USB device in base.txt entering 'yes' here is mandatory. Otherwise the device can't be used. This activates support for USB sticks, external USB drives and USB keyboards as well.

Default setting: `OPT_USB='no'`

USB_EXTRA_DRIVER_N Number of drivers to load.

Default setting: `USB_EXTRA_DRIVER_N='0'`

USB_EXTRA_DRIVER_x Driver that should be loaded.

Possible values at the moment

- printer - support for USB printers
- belkin_sa - USB Belkin Serial converter
- cyberjack - REINER SCT cyberJack pinpad/e-com USB Chipcard Reader
- digi_acceleport - Digi AccelePort USB-2/USB-4 Serial Converter
- empeg - USB Empeg Mark I/II
- ftdi_sio - USB FTDI Serial Converter
- io_edgeport - Edgeport USB Serial
- io_ti - Edgeport USB Serial
- ipaq - USB PocketPC PDA
- ir-usb - USB IR Dongle
- keyspan - Keyspan USB to Serial Converter
- keyspan_pda - USB Keyspan PDA Converter
- kl5kusb105 - KLSI KL5KUSB105 chipset USB->Serial Converter
- kobil_sct - KOBIL USB Smart Card Terminal (experimental)
- mct_u232 - Magic Control Technology USB-RS232 converter
- omninet - USB ZyXEL omni.net LCD PLUS
- pl2303 - Prolific PL2303 USB to serial adaptor
- visor - USB HandSpring Visor / Palm OS

- whiteheat - USB ConnectTech WhiteHEAT

Default setting: `USB_EXTRA_DRIVER_x=”`

USB_EXTRA_DRIVER_x_PARAM Parameters for the driver. Usually you won't need this.

Default setting: `USB_EXTRA_DRIVER_x_PARAM=”`

USB_MODEM_WAITSECONDS Default setting: `USB_MODEM_WAITSECONDS='21'`

Unfortunately Eagle and Speedtouch USB Modems need a long time to get ready. In most cases the default setting of about 21 seconds is sufficient for initialisation. Sometimes the value can be halved and Eagle or Speedtouch USB modem are ready after 10 seconds. If this is the case you could set the value to 10 seconds. If you are unlucky the value has to be raised. Only testing can help you here.

4.22.1. Problems With USB Devices

There may be problems with some USB devices. This can have different causes, for example the driver software or the USB controller.

In the present version the Eagle USB ADSL modem only works when it is also connected to the splitter. If this is not the case the corresponding eth device is not generated. As a result, the modem is not usable. So please connect the modem to the telephone line before.

4.22.2. Hints For Use

It is important to ensure that the hardware USB support is enabled. Especially with onboard USB controllers that is important. For example a WRAP is shipped without USB port. USB is added only by an additional module and is therefore disabled in the BIOS by default.

4.22.3. Mounting Of USB Devices

Plugged USB devices will be detected automatically but must be mounted and unmounted 'by hand'. When plugging an USB stick it is recognized as a SCSI block device. For this reason is accomplished via device sd# for SuperFloppy devices or sd#<Partition-number> for devices with a partition table. USB drives are treated as hard disks and addressed as sda1 and sdb1 if plugged in two USB ports. USB floppies are addressed via sda or sdb without specifying a partition number.

Thus a first USB stick can be mounted to /mnt by executing

mount /dev/sda1 /mnt

A second one at the same would need

mount /dev/sdb1 /mnt

to be mounted. Devices are numbered in the sequence of plugging - first USB device = sda, second USB device = sdb a.s.o. This means that device numbering is not fixed but depending on the sequence of plugin. Unmounting is done via

umount /mnt

If using more that one USB device never mount all devices to the same directoy. Create directories under /mnt for each device to be mounted. Make the directories by executing:

mkdir /mnt/usba
mkdir /mnt/usbb

Then specify the directories as destinations when mounting:

mount /dev/sda1 /mnt/usba
mount /dev/sdb1 /mnt/usbb

The content of the USB devices can be found at /mnt/usba res. /mnt/usbb. Unmounting can be done via

umount /mnt/usba
umount /mnt/usbb

If an USB device has more partitions the device directories under /mnt have to reflect this structure in subdirectories.

4.23. WLAN - Support For Wireless-LAN

When using PCI Cards please be sure to use a mainboard that at least complies to the specifications of PCI 2.2. Older mainboards that only support PCI 2.1 or less can produce diverse errors. Either the computer does not start at all (it even can't be switched on) or the WLAN card is not found on PCI scan.

WLAN cards are adressed as wlanX in base.txt's `IP_NET_X_DEV`. If only one WLAN card is installed its name is wlan0.

4.23.1. WLAN Configuration

OPT_WLAN Default setting: `OPT_WLAN`='no'

Activates package Wireless LAN.

WLAN_WEBGUI Default setting: `WLAN_WEBGUI`='yes'

Aktivates the web interface for package Wireless LAN.

WLAN_REGDOMAIN This variable determines the country settings. Valid values are ISO 3166-1 alpha-2 country codes i.e. 'DE' for Germany. In most countries different legal presets apply for frequency channels and transmission power.

WLAN_N Number of independent WLAN Configurations. If this is set to '1' behavior is like in earlier fli4l versions when only one configuration was allowed.

WLAN_x_MAC MAC address of the WLAN card in this notation:

XX:XX:XX:XX:XX:XX

Each X is a Hex Digit of the MAC Address for the card that belongs to this configuration. If none of the MAC addresses entered here matches a particular card the configuration in `WLAN_1_*` will be applied to this card and a warning message containing the card's MAC address will be displayed. Enter this address in the config file to assure that the web interface will work without problems.

WLAN_x_MAC_OVERRIDE This setting changes the MAC address of the WLAN card. This is used to connect to a WLAN where MAC-filtering is active without changing the filters. This is useful for WAN connections that are fixed to i.e. the MAC address of the WLAN-USB-stick delivered by the provider.

WLAN_x_ESSID The Service Set Identifier (SSID) is the name for your wireless lan. The string has a maximum length of 32 characters and is also called "Network Name". It is configured in the access point of a wireless lan and is used by all clients accessing it. The SSID has to be identical for all joining nodes also for Ad-Hoc networking.

WLAN_x_MODE Sets the WLAN mode to be used by the card.

Default setting: `WLAN_x_MODE`='ad-hoc'

Possible values:

ad-hoc	wireless net without Access-Point
managed	managed wireless net with several cells
master	the WLAN card is working as an Access-Point

`WLAN_x_MODE`='master' will only working with adequate WLAN drivers.

WLAN_x_NOESSID Deactivates sending ESSID during beacon frames. Only valid with hostap_* driver and Firmware >= 1.6.3 in WLAN_MODE='master'

This feature is optional and has to be added manually in config/wlan.txt.

WLAN_x_CHANNEL Sets the transmission channel of the network.

Default setting: `WLAN_x_CHANNEL`='1'

Possible values: 1-13 and 36,40,44,48,52,56,60,64,100,104,108,112,116,120,124,128,132, 136,140

Please read the documentation of your WLAN card to find out which channels are allowed to use in your country. You are responsible for any rights violation by using channels not allowed. In Germany channels 1-13 in 2,4 GHz frequency range (Modes: b and g) are allowed. Channels 36-140 (see above) are legally allowed in 5 GHz frequency range.

Value '0' is also valid if `WLAN_x_MODE`='managed' is set. This does not set a particular channel explicitely but searches for an AP an all valid channels. You may add the character a,b or g to the channel (i.e. 5g) which will then determine the operating mode and frequency range.

Adding 'n' or 'N' triggers usage of 802.11n for according WLAN cards. Lower case means: 20 MHz channel width, upper Case means: 40 MHz channel width.

Upper case for a/b/g will activate proprietary WLAN turbo modes with some drivers (at the moment only ath_pci). This option is experimental and may be removed at any time.

WLAN_x_RATE Sets transmission speed of the network.

Default setting: `WLAN_x_RATE`='auto'

Possible values: 1,2,5.5,11,auto - rates in Megabit/s
Depending on the card this rates can be chosen in addition: 6,9,12,18,24,36,48 and 54. Some 54 MBit cards don't accept rate settings. In this case specify 'auto' here.

WLAN_x_RTS Activates RTS/CTS handshake. This option is useful in bigger WLANs with a lot of clients if those clients can't receive each other but only the AP. If this option is activated the client will start each transmission with a RTS query to get permission for the actual data transmission. It gets a CTS from the AP then providing permission to send. This way every client knows another client is transmitting without receiving the other client. Collisions are minimized because it is ensured that only one client is transmitting data at a time. This option only makes sense in the situation described above because of additional overhead and thus decreasing bandwidth. Bandwidth can raise by avoiding collision though.

This feature is optional and has to be added manually in config/wlan.txt.

WLAN_x_ENC_N (deprecated) Sets the number of Wireless Encryption Key's (WEP).

Possible values: 0-4

WLAN_x_ENC_x (deprecated) Sets Wireless Encryption Keys.

Possible values:

XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XX	128 Bit Hex-Key (X=0-F)
XXXX-XXXX-XX	64 Bit Hex-Key (X=0-F)
s:<5 characters>	64 Bit
s:<6-13 characters>	128 Bit
P:<1-64 characters>	128 Bit

Using s:text is **not** compatible with the passphrase of the Windows drivers. Use a hex key instead! Windows mostly uses hex keys **without** hyphens '-'. Using P:<text> is compatible to passphrases of most (if not all) Windows WLAN drivers but **only** in 128 Bit mode. Linux allows to mix key length which Windows drivers usually do **not**!

WLAN_x_ENC_ACTIVE (deprecated) Sets the active wireless encryption key.

Possible values: 1-4

This variable must be set if `WLAN_x_ENC_N` is greater than 0. In other cases it's optional.

WLAN_x_ENC_MODE (deprecated) Activates the encryption mode.

Possible values:

on/off	with or without encryption
open	also accepts unencrypted packets
restricted	only accepts encrypted packets

Most reasonable value: 'restricted'

This feature is optional and has to be added manually in config/wlan.txt. If this variable is not set the default 'off' will be assumed if no WEP key was defined and 'restricted' if at least one key is defined.

WLAN_x_WPA_KEY_MGMT If you want wo use WPA instead of WEP encryption set the WPA mode here. At the moment only WPA with a pre-shared key between client and AP (WPA-PSK) is supported. This key should be chosen carefully and not too short because that would allow for dictionary attacks.

In *managed* mode all cards supported by WPA-Supplicant (http://hostap.epitest.fi/wpa_supplicant/ and in *master* mode all cards supported by Hostapd (http://hostap.epitest.fi/hostapd/) are accepted.

Cards based on chipsets by Atheros and on the hostap-driver were tested successfully in managed and master mode. In theory also atmel and some other cards are possible if developers of third-party opt-packages adapt their packages accordingly.

WLAN_x_WPA_PSK Specify the pre-shared key to be used for communication between client and Access-Point here. The key is provided as a passphrase at a minimum length of 16 characters and a maximum length of 63 characters. The following characters are supported:

a-z A-Z 0-9 ! # $ % & ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~

WLAN_x_WPA_TYPE Choose between '1' for WPA1 mode, '2' for WPA2 (IEEE 802.11i) mode and '3' for both - the client can decide to use WPA1 or WPA2. Only WPA2 should be used if the hardware supports it.

WLAN_x_WPA_ENCRYPTION Encryption protocols TKIP and the enhanced version CCMP (AES-CTR/CBC-MAC Protocol, mostly called AES in short) can be used. CCMP eventually won't be supported by older WLAN hardware. You may also specifiy both.

WLAN_x_WPA_DEBUG In case of problems with WPA set this variable to 'yes' for the daemon to provide more verbose output for debugging.

WLAN_x_AP Registers the node with an Access-Point.

Specifiy the MAC address of the Access-Points here. If WLAN mode "master" was chosen before keep this variable empty. This option only makes sense if fli4l can't find the AP by itself or should be bound to a preferred Access-Point. Only to be used in WLAN mode "managed".

This feature is optional and has to be added manually in config/wlan.txt.

WLAN_x_ACL_POLICY Access Control List (ACL) Policy.

Default setting: `WLAN_x_ACL_POLICY='allow'`

Describes what action should be taken for the provided MAC addresses:

deny	None of the addresses listed here gets access
allow	Only the addresses listed here get access
open	All MAC addresses get access independent on filter

Unfortunately WLAN_ACLs are only supported well by the hostap_* driver. As an alternative you may use the firewall rules that have improved a lot since fli4l version 3.0.x.

WLAN_x_ACL_MAC_N Number of restricted WLAN stations.

Default setting: `WLAN_x_ACL_MAC_N='0'`

A number greater than 0 activates the Access Control List.

WLAN_x_ACL_MAC_x MAC address in notation XX:XX:XX:XX:XX:XX (example: 00:00:E8:83:72:92)

WLAN_x_DIVERSITY Trigger manual antenna diversity.

Default setting: `WLAN_x_DIVERSITY='no'` (automatic)

WLAN_x_DIVERSITY_RX The receiving antenna to be used.

Default setting: `WLAN_x_DIVERSITY_RX='1'`

0 = Automatic
1 = Antenna 1
2 = Antenna 2

WLAN_x_DIVERSITY_TX The transmitting antenna to be used.

Default setting: `WLAN_x_DIVERSITY_TX='1'`

WLAN_x_WPS Activates WPS support. Push-Button and PIN are possible. If you don't only want to use the console it makes sense to activate WLAN_WEBGUI.

Default setting: `WLAN_x_WPS='no'`

WLAN_x_PSKFILE With PSKFILE activated other client related keys can be used beside `WLAN_x_WPA_PSK` pre-shared keys. At the moment the function `WLAN_x_WPS` uses this file to provide individual keys to clients.

If the file is deactivated WPS clients using it can not connect to the Access Point anymore.

WPS-Clients connected with deactivated file are not affected.

Default setting: `WLAN_x_PSKFILE='yes'`

WLAN_x_BRIDGE As an alternative to package ADVANCED_NETWORKING you may specifiy the bridge to which the WLAN should be bound here.

Example: `WLAN_x_BRIDGE='br0'`

Attention: Use either ADVANCED_NETWORKING or this setting and **not both**!

4.23.2. Examples

Connecting to an Access Point via WPA

```
OPT_WLAN='yes'
WLAN_N='1'
WLAN_1_MAC='00:0F:A3:xx:xx:xx'
WLAN_1_ESSID='foo'
WLAN_1_MODE='managed'           # connection to an Access Point
WLAN_1_CHANNEL='1'
WLAN_1_RATE='auto'
#
```

```
# WPA Configuration
#
WLAN_1_ENC_N='0'                # no WEP
WLAN_1_WPA_KEY_MGMT='WPA-PSK'   # WPA pre shared key
WLAN_1_WPA_TYPE='1'             # WPA 1
WLAN_1_WPA_ENCRYPTION='TKIP'
WLAN_1_WPA_PSK='your best passphrase choice ever (16-63 characters)'
#
# irrelevant in WPA context
#
WLAN_1_ENC_N='0'
WLAN_1_ENC_ACTIVE='1'
WLAN_1_ACL_POLICY='allow'
WLAN_1_ACL_MAC_N='0'
```

Access Point with WPA2 Encryption

```
OPT_WLAN='yes'
WLAN_N='1'
WLAN_1_MAC='00:0F:A3:xx:xx:xx'
WLAN_1_ESSID='foo'
WLAN_1_MODE='master'            # Access Point
WLAN_1_CHANNEL='1g'             # Channel 1, Mode 'g' on an
                                # Atheros card
WLAN_1_RATE='auto'
#
# WPA Configuration
#
WLAN_1_ENC_N='0'                # no WEP
WLAN_1_WPA_KEY_MGMT='WPA-PSK'   # WPA pre shared key
WLAN_1_WPA_TYPE='2'             # WPA 2
WLAN_1_WPA_ENCRYPTION='CCMP'
WLAN_1_WPA_PSK='your best passphrase choice ever (16-63 characters)'
#
# MAC based Access Control to AP
#
WLAN_1_ACL_POLICY='allow'
WLAN_1_ACL_MAC_N='0'
#
# irrelevant in WPA context
#
WLAN_1_ENC_ACTIVE='1'
```

Access Point with WEP Encryption

```
OPT_WLAN='yes'
WLAN_N='1'
WLAN_1_MAC='00:0F:A3:xx:xx:xx'
WLAN_1_ESSID='foo'
WLAN_1_MODE='master'            # Access Point
WLAN_1_CHANNEL='1'
WLAN_1_RATE='auto'
#
```

```
# WEP Configuration
#
WLAN_1_WPA_KEY_MGMT=''          # no WPA
WLAN_1_ENC_N='4'                # 4 WEP-Keys
WLAN_1_ENC_1='...'
WLAN_1_ENC_2='...'
WLAN_1_ENC_3='...'
WLAN_1_ENC_4='...'
WLAN_1_ENC_ACTIVE='1'           # first key is active
#
# MAC based Access Control to AP
#
WLAN_1_ACL_POLICY='allow'
WLAN_1_ACL_MAC_N='0'
#
# irrelevant for WEP Configuration
#
WLAN_1_WPA_TYPE='2'
WLAN_1_WPA_ENCRYPTION='CCMP'
WLAN_1_WPA_PSK='...'
```

4.23.3. Virtual Accesspoint (VAP) (experimental)

Certain WLAN cards (driver: ath_pci, ath5k, ath9k, ath9k_htc) can be split into a maximum of 4 virtual WLAN cards. (VAP)

WLAN configuration of virtual APs is arbitrary except for one condition: Channel and MAC address have to be the same. Based on the multiplexed MAC address the card that should be splitted is identified. If more physical cards are present this can be done repeatedly for each of them.

The base device's name is still wlan0, the next in VAP mode will be wlan0v2 a.s.o. For binding to a bridge please use WLAN_x_BRIDGE='br0'!

The maximum at the moment is up to 8x master depending on card and driver.

4.23.4. Switching WLAN on and off based on daytime with easycron

By using the package *easycron* (Page 115) WLAN may be switched on and off based on a time schedule.

```
EASYCRON_N='2'
EASYCRON_1_CUSTOM  = ''     # switch off every evening at 12PM
EASYCRON_1_COMMAND = '/usr/sbin/wlanconfig.sh wlan0 down'
EASYCRON_1_TIME    = '* 24 * * *'

EASYCRON_2_CUSTOM  = ''     # and  on at 8 AM.
EASYCRON_2_COMMAND = '/usr/sbin/wlanconfig.sh wlan0'
EASYCRON_2_TIME    = '* 8 * * *'
```

4.23.5. Donations

Due to a generous donation of 2 Ralink 2500 based WLAN cards with RT25xx chipset those cards can be used in ad-hoc and managed modes. Use rt2500 as the driver in base.txt.

The cards where donated by:
Computer Contor, Pilgrimstein 24a, 35037 Marburg marklabelbuildroot

4.24. SRC - The fli4l Buildroot

This chapter is mainly of interest for developers, who want to compile binaries or the Linux kernel for the fli4l. If you only want to use flil4 as a router and do not offer packages for the fli4l needing own binaries, you may skip this chapter completely.

In general, for compiling program packages for the fli4l a Linux system is required. Compilation under other operating systems (Microsoft Windows, OS X, FreeBSD, etc.) is *not* supported.

The requirements for a Linux system for fli4l development are as follows:

- GNU gcc and g++ in version 2.95 or higher
- GNU gcc-multilib (depending on your host system)
- GNU binutils (contains bind and other necessary programs)
- GNU make in version 3.81 or higher
- GNU bash
- libncurses5-dev for `fbr-make *-menuconfig` (depending on your host system)
- The programs sed, awk, which, flex, bison and patch
- The programs makeinfo (package texinfo) and msgfmt (package gettext)
- The programs tar, cpio, gzip, bzip2 and unzip
- The programs wget, rsync, svn and git
- The programs perl and python

In the following, characters printed **bold** represent keyboard input, the ↵-character stands for the Enter key on your keyboard and executes entered commands.

4.24.1. The Sources - An Overview

In the directory `src` you will find the following subdirectories:

Directory	Content
`fbr`	In this folder there is a custom build system based on the buildroot for uClibc (currently in version 0.9.33.2). FBR stands for "flil4-Buildroot". It is thus possible, to compile all programs used on fli4l (kernel, libraries and applications) anew.

Directory	Content
`fli4l`	This directory contains the fli4l specific sources for packages sorted by their names. All sources that are included in this subdirectory, were either written specifically for use with fli4l or are at least strongly adapted.
`cross`	This directory contains scripts that create the cross-compilers necessary for compiling mkfli4l for various Platforms.

4.24.2. Compile A Program For fli4l

In the subdirectory "fbr" a script `fbr-make` exists which controls the compilation of all the programs from the basic packages for fli4l. This script takes care of downloading and compiling all required binaries for fli4l. The script will save files in the directory `~/.fbr`, if it does not yet exist, it will be created. (The directory may be changed by using the environment variable `FBR_BASEDIR`, see below.)

Note: During the compilation process much space is needed (currently around 900 MiB for the downloaded archives and almost 30 GiB for the intermediate results and the resulting compiled files). Hence, make sure to have enough space under `~/.fbr`! (Alternatively, you may also use the `FBR_TIDY` option, see below.)

The directory structure below `~/.fbr` is as follows:

Directory	Content
`fbr-<branch>-<arch>`	The uClibc buildroot is unpacked here. `<branch>` stands for the development branch the FBR is derived from, in this case (i.e. `trunk`). If the origin of the FBR is an unpacked `src` package, `fbr-custom` will be used. `<arch>` reflects the processor architecture in use (i.e. `x86` or `x86_64`). More concerning this directory can be found below.
`dl`	Here the downloaded archives are stored.
`own`	Here own FBR packages which should be compiled can be stored.

Under the Buildroot directory `~/.fbr/fbr-<branch>-<arch>/buildroot` the following directories are of interest:

Directory	Content
`output/sandbox`	In this directory a subdirectory exists for each FBR package that holds the files of the package after being compiled. In the directory `output/sandbox/<package>/target` the files for the fli4l router can be found in this case. In the directory `output/sandbox/<package>/staging` files needed for building *other* FBR packages requiring this FBR package can be found.

Directory	Content
`output/target`	In this directory *all* compiled programs for the fli4l-router are stored. This directory mirrors the directory structure on the fli4l router. By the help of `chroot` you can change to this directory and try the programs.[15]

General Settings

The operation of `fbr-make` can be influenced by various environment variables:

Variable	Description
`FBR`	Specifies the path to the FBR explicitely. Per default the path `~/.fbr/fbr-<branch>-<arch>` (see above) is used.
`FBR_BASEDIR`	explicitely specifies the base path to the FBR. As a default the `~/.fbr` (see above) will be used. This variable will be ignored if the environment variable `FBR` is set.
`FBR_DLDIR`	Specifies the directory containing the source archives. Per default the path `${FBR}/../dl` (i.e. `~/.fbr/dl`) will be used.
`FBR_OWNDIR`	Specifies the directory holding the own packages. Per default the path `${FBR}/../own` (i.e. `~/.fbr/own`) will be used.
`FBR_TIDY`	if this variable is set to "y" intermediate results while compiling the FBR packages will be deletetd immediately after their installation to the directory `output/target`. This saves a lot of space and is always recommended if you do not feel the urge to have a look at `output/build/...` after the build process. If this variable 'contains the value " k', only the intermediate results in the various Linux kernel directories are removed, because this saves a lot of space without losing any functionality. All other assignments (or if the variable is missing entirely) ensure that all intermediate results are kept.
`FBR_ARCH`	This variable specifies the processor architecture for which the FBR (or FBR packages) should be built. If it is missing, `x86` will be used. The supported architectures can be found below.

The FBR currently supports the following architectures:

Architecture	Description
`x86`	Intel x86-Architecture (32-Bit), also known as IA-32.
`x86_64`	AMD x86-64-Architecture (64-Bit), also called Intel 64 or EM64T by Intel.

[15] This is bound to some preconditions, see the paragraph "Testing Of A Compiled Program" (Page 242).

Compilation Of All FBR Packages

If executing `fbr-make` with the argument `world`, it may last several hours to download and compile all source archives, depending on the computer and Internet connections used. [16]

Compiling The Toolchain

If executing `fbr-make` with the argument `toolchain`, all FBR packages needed for building the fli4l binary programs will be downloaded and compiled (Compiler, Binder, uClibc-library etc.). Normally this command is not needed, because all FBR packages depend on the toolchain and thus it has to be downloaded and build anyway.

Compiling A Single FBR Package

If you only want to compile a certain FBR package (i.e. the programs of an OPT developed yourself), you may transfer the name of one or more FBR packages to the `fbr-make` program (for example `fbr-make openvpn` to download and compile the OpenVPN programs). All dependencies will also be downloaded and compiled.

Recompilation Of A Single FBR Package

If you like to compile a FBR package again (for whatever reason), you first need to remove the information on the FBR about the previous compilation process. For this purpose use the command `fbr-make <package>-clean` (eg `fbr-make openvpn-clean`). In this case informations about all dependencies for the package are also reset causing their recompilation with the next `fbr-make world` as well.

Recompiling All FBR Packages

If you like to recompile the entire FBR (eg because you want to use it as a benchmark program for your new high-end developer system ;-), you can use the command `fbr-make clean` and remove all artifacts that have been generated during the last FBR build. You will have to confirm this action. [17] This is also useful to free used disk space.

4.24.3. Testing Of A Compiled Program

If a program has been compiled with `fbr-make` it may also be tested on the development machine. Such a test will of course only work if the processor architecture of the developer machine matches the processor architecture the fli4l programs were compiled for. (It is not possible, for example, to run `x86_64` flil4 programs on a `x86` operating system.) If this condition is met, change to the fli4l target directory with

```
chroot ~/.fbr/fbr-<branch>-<arch>/buildroot/output/target /bin/sh↵
```

and test the compiled program(s) there immedeately. Please note that executing `chroot` needs administrator rights and thus you have to use `sudo` or `su`, depending on preference and system

[16] Downloading of source archives is of course done only once as long as you don't update the FBR and thus need new package versions and source archives.

[17] The whole directory `~/.fbr/fbr-<branch>-<arch>/buildroot/output` will be removed.

configuration! In addition you will have to compile the FBR package `busybox` (via `fbr-make busybox`), to have a working shell in the `chroot` environment. A small example:

```
$ sudo chroot ~/.fbr/fbr-trunk-x86/buildroot/output/target /bin/sh↵
Passwort:(Your password)↵


BusyBox v1.22.1 (fli4l) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# ls↵
THIS_IS_NOT_YOUR_ROOT_FILESYSTEM  mnt
bin                               opt
dev                               proc
etc                               root
home                              run
img                               sbin
include                           share
lib                               sys
lib32                             tmp
libexec                           usr
man                               var
media                             windows
# bc --version↵
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
# echo "42 - 23" | bc↵
19
#
```

4.24.4. Debugging Of A Compiled Program

In case of problems with compiled fli4l programs (crashes) you have the option to analyze the state of the program immediately before the crash (also called "post-mortem debugging"). To do so, activate `DEBUG_ENABLE_CORE='yes'` in the configuration of the `base` package. If case of a crash a memory dump is generated in `/var/log/dumps/core.<PID>`. "PID" is the process ID of the crashed process. You may analyze the state of the program on a Linux machine with a fully compiled FBR as described below. The following example to be analyzed is the program `/usr/sbin/collectd`, which was terminated with a SIGBUS. The dump was stored in `/tmp/core.collectd`.

```
fli4l@eisler:~$ .fbr/fbr-trunk-x86/buildroot/output/host/usr/bin/i586-linux-gdb↵
GNU gdb (GDB) 7.5.1
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=x86_64-unknown-linux-gnu --target=i586-buildr
oot-linux-uclibc".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
```

```
(gdb) set sysroot /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target↵
(gdb) set debug-file-directory /project/fli4l/.fbr/fbr-trunk-x86/buildroot/outpu
t/debug↵
(gdb) file /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target/usr/sbin/co
llectd↵
Reading symbols from /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target/u
sr/sbin/collectd...Reading symbols from /project/fli4l/.fbr/fbr-trunk-x86/buildr
oot/output/debug/.build-id/8b/28ab573be4a2302e1117964ededе2e54ebbdbf.debug...don
e.
done.
(gdb) core /tmp/core.collectd↵
[New LWP 2250]
[New LWP 2252]
[New LWP 2259]
[New LWP 2257]
[New LWP 2255]
[New LWP 2232]
[New LWP 2235]
[New LWP 2238]
[New LWP 2242]
[New LWP 2244]
[New LWP 2245]
[New LWP 2231]
[New LWP 2243]
[New LWP 2251]
[New LWP 2248]
[New LWP 2239]
[New LWP 2229]
[New LWP 2249]
[New LWP 2230]
[New LWP 2247]
[New LWP 2233]
[New LWP 2256]
[New LWP 2236]
[New LWP 2246]
[New LWP 2240]
[New LWP 2241]
[New LWP 2237]
[New LWP 2234]
[New LWP 2253]
[New LWP 2254]
[New LWP 2258]
[New LWP 2260]
Failed to read a valid object file image from memory.
Core was generated by `collectd -f'.
Program terminated with signal 7, Bus error.
#0  0xb7705f5d in memcpy ()
   from /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target/lib/libc.so.0
(gdb) backtrace↵
#0  0xb7705f5d in memcpy ()
   from /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target/lib/libc.so.0
#1  0xb768a251 in rrd_write (rrd_file=rrd_file@entry=0x808e930, buf=0x808e268,
    count=count@entry=112) at rrd_open.c:716
```

```
#2  0xb76834f3 in rrd_create_fn (
    file_name=file_name@entry=0x808d2f8 "/data/rrdtool/db/vm-fli4l-1/cpu-0/cpu-i
nterrupt.rrd.async", rrd=rrd@entry=0xacff2f4c) at rrd_create.c:727
#3  0xb7683d7b in rrd_create_r (
    filename=filename@entry=0x808d2f8 "/data/rrdtool/db/vm-fli4l-1/cpu-0/cpu-int
errupt.rrd.async", pdp_step=pdp_step@entry=10, last_up=last_up@entry=1386052459,
    argc=argc@entry=16, argv=argv@entry=0x808cf18) at rrd_create.c:580
#4  0xb76b77fd in srrd_create (
    filename=0xacff33f0 "/data/rrdtool/db/vm-fli4l-1/cpu-0/cpu-interrupt.rrd.asy
nc",
    pdp_step=10, last_up=1386052459, argc=16, argv=0x808cf18) at utils_rrdcreate
.c:377
#5  0xb76b78cb in srrd_create_thread (targs=targs@entry=0x808bab8)
    at utils_rrdcreate.c:559
#6  0xb76b7a8f in srrd_create_thread (targs=0x808bab8) at utils_rrdcreate.c:491
#7  0xb7763430 in ?? ()
   from /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target/lib/libpthread
.so.0
#8  0xb775e672 in clone ()
   from /project/fli4l/.fbr/fbr-trunk-x86/buildroot/output/target/lib/libpthread
.so.0
(gdb) frame 1↵
#1  0xb768a251 in rrd_write (rrd_file=rrd_file@entry=0x808e930, buf=0x808e268,
    count=count@entry=112) at rrd_open.c:716
716         memcpy(rrd_simple_file->file_start + rrd_file->pos, buf, count);
(gdb) print (char*) buf↵
$1 = 0x808e268 "RRD"
(gdb) print rrd_simple_file->file_start↵
value has been optimized out
(gdb) list↵
711         if((rrd_file->pos + count) > old_size)
712         {
713             rrd_set_error("attempting to write beyond end of file");
714             return -1;
715         }
716         memcpy(rrd_simple_file->file_start + rrd_file->pos, buf, count);
717         rrd_file->pos += count;
718         return count;       /* mimmic write() semantics */
719     #else
720         ssize_t   _sz = write(rrd_simple_file->fd, buf, count);
(gdb) list 700↵
695      * rrd_file->pos of rrd_simple_file->fd.
696      * Returns the number of bytes written or <0 on error.  */
697
698     ssize_t rrd_write(
699         rrd_file_t *rrd_file,
700         const void *buf,
701         size_t count)
702     {
703         rrd_simple_file_t *rrd_simple_file = (rrd_simple_file_t *)rrd_file->
pvt;
704     #ifdef HAVE_MMAP
(gdb) print *(rrd_simple_file_t *)rrd_file->pvt↵
```

```
$2 = {fd = 9, file_start = 0xa67d0000 <Address 0xa67d0000 out of bounds>,
  mm_prot = 3, mm_flags = 1}
```

After some "digging", you will see that an invalid pointer is contained in the `rrd_simple_file_t` object ("Address ... out of bounds"). In the further process of debugging, it became clear that a failed `posix_fallocate`-call was the culprit for the crash.

It is important to pass *all* paths fully qualified (`/project/...`) and to use no "shortcuts" (i.e. in `~/...`). If you don't obey this it may happen that `gdb` will not find the debug informations for the application and/or for the libraries in use. Due to space reasons debug informations are not contained directly in the program to be investigated but are saved in separate files in the directoy `~/.fbr/fbr-<branch>-<arch>/buildroot/output/debug/`.

4.24.5. Informations On The FBR

Displaying Help

Use the command `fbr-make help` to see what `fbr-make` can do for you.

Displaying Program Informations

By using the command `fbr-make show-versions` you can review all FBR packages provided with version number:

```
$ fbr-make show-versions↵
Configured packages

acpid 2.0.20
actctrl 3.25+dfsg1
add-days undefined
[...]
```

Display Of Dependencies On Libraries

With `fbr-make links-against <soname>` all files in `~/.fbr/fbr-<branch>-<arch>/buildroot/output/target` linked to a library named `soname` can be shown. If for example all programs and libraries should be identified that use `libm` (Library with mathematical functions) use `fbr-make links-against libm.so.0` because `libm.so.0` is the name of the libm library. A possible output would be:

```
$ fbr-make links-against librrd_th.so.4↵
Executing plugin links-against
Files linking against librrd_th.so.4
collectd usr/lib/collectd/rrdcached.so
collectd usr/lib/collectd/rrdtool.so
rrdtool usr/bin/rrdcached
```

In the first column is the package name and in the second the (relative) path to the file that is linked against the library in question.

To find the library name for a library, you can use `readelf` like this:

```
$ readelf -d ~/.fbr/fbr-trunk-x86/buildroot/output/target/lib/libm-0.9.33.2.so |↵
> grep SONAME↵
 0x0000000e (SONAME)                     Library soname: [libm.so.0]
```

Display Of Version Changes

The command `fbr-make version-changes` is interesting (only) for fli4l-team developers with write access to the fli4l SVN repository. It lists all FBR packages whose version has been modified locally, i.e. those where the version in the working copy differs from the repository version. This helps the developer to get an overview on updated FBR packages before writing the changes to the repo. A possible output is:

```
$ fbr-make version-changes↵
Executing plugin version-changes
Package version changes
KAMAILIO: 4.0.5 --> 4.1.1
```

Here you can see that the package `kamailio` in FBR was updated from version 4.0.5 to version 4.1.1.

4.24.6. Changing The FBR Configuration

Reconfiguration Of The FBR

By using `fbr-make buildroot-menuconfig` it is possible to select the FBR packages to be compiled. This is useful if you want to compile other FBR packages for the fli4l that are not enabled by default but are integrated in the uClibc buildroot, or if you want to activate own FBR packages. On the other other hand global properties of FBR may be changed, such as the version of the used GCC compiler. On successful exit of the configuration menu, the new configuration is saved in the directory `src/fbr/buildroot/.config`.

Please note, however, that such changes of the toolchain configuration are *not* officially supported because the resulting binaries will be incompatible with the official fli4l distribution with a high probability. So if you need binaries for your own OPT and want to publish this OPT, you should not change the toolchain settings!

Reconfiguration Of The uClibc Library

With `fbr-make uclibc-menuconfig` the funcionality of the uClibc library in use may be changed. On successful exit of the configuration menu, the new configuration is saved to `src/fbr/buildroot/package/uclibc/uclibc.config`.

Like in the last paragraph also here applies: A change is most likely not compatible with the official fli4l distribution and is thus not supported!

Reconfiguration Of Busybox

With `fbr-make busybox-menuconfig` the Busybox may be changed in its funcionality. On successful exit of the configuration menu, the new configuration is saved to `src/fbr/buildroot/package/busybox/busybox-<Version>.config`.

Also here applies: A change is most likely not compatible with the official fli4l distribution and is thus not supported! Adding new Busybox applets causes no

problems as long as you only use the modified Busybox on your own flil4 router (and not require the users of your OPT to use a Busybox modified in this way).

Reconfiguration Of The Linux Kernel Packages

With `fbr-make linux-menuconfig` resp. `fbr-make linux-<version>-menuconfig` the configuration of all activated Kernel packages resp. a specific Kernel package may be changed. On successful exit of the configuration menu, the new configuration is saved to `src/fbr/buildroot/linux/linux-<version>/dot-config-<arch>`. [18]

Like in the last paragraph also here applies: A change is most likely not compatible with the official fli4l distribution and is thus not supported! At most, adding of new modules to the Linux kernel is easy, as long you only use the modified kernel on your own flil4 router (and not require the users of your OPT to use a Kernel modified in this way).

4.24.7. Updating The FBR

Each of the commands outlined above is advanced by an an examination of the actuality of the FBR. If a discrepancy between the sources in which `fbr-make` is located (unpacked `src`-package or SVN-working copy) and the FBR in `~/.fbr/fbr-<branch>-<arch>/buildroot` is detected the latter will be updated. New FBR packages will be integrated and old, no longer contained FBR packages will be deleted. The configurations are compared: FBR packages with modified configuration and all dependent FBR packages will be rebuilt. This ensures that the FBR on your computer is always equal to the developer's one (except for your own FBR packages under `~/.fbr/own/`). **However, This also means that changes to the official part of the Buildroot configuration will be lost with the next update!** Therefore it is not recommended to reconfigure FBR, at least not if you are using `src` packages instead of a SVN working copy. (When updating a SVN working copy your local configuration changes and the changes to the SVN repository will be merged and the problem of lost configuration does not occur.) However, your own FBR packages may be reconfigured easily, without data loss occuring on an update.

4.24.8. Integrating Own Programs Into The FBR

Compilation of the individual FBR packages is controlled by small Makefiles. If you want to develop your own FBR packages, you have to create a Makefile and a configuration description in `~/.fbr/own/<package>/`. How these are constructed and how to write own Makefiles derived from them is decribed in detail in the documentation for the uClibc Buildroot http://buildroot.uclibc.org/downloads/manual/manual.html#adding-packages.

[18] This only applies to the standard Kernel. For variants of a Kernel package a `diff` file will be saved in `src/fbr/buildroot/linux/linux-<version>/linux-<version>_<variante>/dot-config-<arch>.diff` instead.

5. Creating the fli4l Archives/Boot media

If all configuration is completed, the fli4l archives/boot media may be created as either bootable Compact-Flash, a bootable ISO image, or only the files needed for a remote update.

5.1. Creating the fli4l Archives/Boot media under Linux or other Unix derivatives and Mac OS X

This is done by using scripts (`.sh`), which can be found in the fli4l root directory.

`mkfli4l.sh`

The Build Script recognizes the different boot types (Page 24).
The simplest call on Linux looks like this:

```
sh mkfli4l.sh
```

The actions of the build scripts are controlled by three mechanisms:

- Configuration variable `BOOT_TYPE` from `<config>/base.txt`
- Configuration file `<config>/mkfli4l.txt`
- Build-Script Parameters

The variable `BOOT_TYPE` (Page 24) decides which action of the Build Scripts is executed:

- Create a bootable fli4l CD-ISO-Image
- Generating the fli4l files needed for a remote update
- Generating the fli4l files and directly do a remote update via SCP
- a.s.o.

The description of the variables in the configuration file `<config>/mkfli4l.txt` can be found in Chapter Control file mkfli4l.txt (Page 257).

5.1.1. Command line options

The last control mechanism is appending option parameters to the call of the Build Script on the command line. The control options correspond to those in the file `mkfli4l.txt`. Option parameters override the values from the control file. Out of convenience, the names of the option parameters differ from the names of the variables from the control file. There is a long and, to some extent, a short form:

```
Usage: mkfli4l.sh [options] [config-dir]

-c, --clean         cleanup the build-directory
-b, --build <dir>   set build-directory to <dir> for the fli4l-files
-h, --help          display this usage
--batch             don't ask for user input

config-dir          set other config-directory - default is "config"

--hdinstallpath <dir> install a pre-install environment directly to
                    usb/compact flash device mounted or mountable to
                    directory <dir> in order to start the real installation
                    process directly from that device
                    device either has to be mounted and to be writable
                    for the user or it has to be mountable by the user
                    Do not use this for regular updates!

*** Remote-Update options
--remoteupdate        remote-update via scp, implies "--filesonly"
--remoteremount       make /boot writable before copying files and
                      read only afterwards
--remoteuser <name>   user name for remote-update - default is "fli4l"
--remotehost <host>   hostname or IP of remote machine - default
                      is HOSTNAME set in [config-dir]/base.txt
--remotepath <path>   pathname on remote maschine - default is "/boot"
--remoteport <portnr> portnumber of the sshd on remote maschine

*** Netboot options (only on Unix/Linux)
--tftpbootpath <path>   pathname to tftpboot directory
--tftpbootimage <name>  name of the generated bootimage file
--pxesubdir <path>      subdirectory for pxe files relative to tftpbootpath

*** Developer options
-u, --update-ver    set version to <fli4l_version>-rev<svn revision>
-v, --verbose       verbose - some debug-output
-k, --kernel-pkg    create a package containing all available kernel
                    modules and terminate afterwards.
                    set COMPLETE_KERNEL='yes' in config-directory/_kernel.txt
                    and run mkfli4l.sh again without -k to finish
    --filesonly     create only fli4l-files - do not create a boot-media
    --no-squeeze    don't compress shell scripts
    --rebuild       rebuild mkfli4l and related tools; needs make, gcc
```

An HD pre-installation of a suitably formatted (FAT16/FAT32) CompactFlash (in a USB cardreader) or an USB Stick can be done by using the option `--hdinstallpath <dir>`. You

are using this script *at your own risk*. The necessary fli4l files will be copied onto the specified partition. At first, run in the fli4l directory:

```
sh mkfli4l.sh --hdinstallpath <dir>
```

This will generate the fli4l files and copy them to the CF-Card or USB Stick.

To run the next steps, you have to make sure:

- `chmod 777 /dev/brain`
- superuser rights
- installed `syslinux`
- installed `fdisk`

The script will ensure that this storage device is a FAT-partitioned USB-Drive. After that the boot loader and the files needed will be copied to the disk. You will get notified about success or failure.

After the build you have to execute the following:

```
syslinux --mbr /dev/brain

# make partition bootable using fdisk
#     p - print partitions
#     a - toggle bootable flag, specify number of fli4l partition
#         usually '1'
#     w - write changes and quit
fdisk /dev/brain

# install boot loader
syslinux -i /dev/brain
```

Now the CF resp. USB-drive should be bootable. Don't forget to unmount the device (via `umount`).

An alternative configuration directory can be specified by appending its name to the end of the command line. The normal configuration directory is called `config` and cn be found under the fli4l root directory. This is where all fli4l packages place their configuration files. If you want to maintain more than one configuration, create another directory, i.e. `hd.conf`, place a copy of the configuration files there and change them according to the requirements. Here are some examples:

```
sh mkfli4l.sh --filesonly hd.conf
sh mkfli4l.sh --no-squeeze config.test
```

5.2. Creating the fli4l Archives/Boot media under Windows

Utilize the tool 'AutoIt3' (http://www.autoitscript.com/site/autoit/). This enables a 'graphical' edition, as well as dialogues which allow to change the variables described in the following sections.

`mkfli4l.bat`

The Build program automatically recognizes the different boot types (Page 24).

The 'mkfli4l.bat' can be invoked directly from Windows Explorer, if you need no optional parameters.

The actions of the Build program are controlled by different mechanisms:

- Configuration variable `BOOT_TYPE` from the `<config>/base.txt`
- Configuration file `<config>/mkfli4l.txt`
- Parameter of the build program
- Interactive settings in the GUI

The variable `BOOT_TYPE` (Page 24) decides which action the Build program executes:

- Create a bootable fli4l CD-ISO-Image
- Making the fli4l files available, for remote update
- Generating the fli4l files and direct remote update via SCP
- Hard drive pre-install of a suitably formatted CF in the Cardreader
- a.s.o.

The description of the variables in the configuration file `<config>/mkfli4l.txt` can be found in Chapter Control file mkfli4l.txt (Page 257).

5.2.1. Command line options

The last control mechanism is appending of option parameters to the call of the Build program on the command line. The control options correspond to those in the control file `mkfli4l.txt`. Option parameters override the values from the control file. Out of convenience, the names of the option parameters differ from the names of the variables from the control file. There is a long and, to some extent, a short form:

```
Usage: mkfli4l.bat [options] [config-dir]

-c, --clean             cleanup the build-directory
-b, --build <dir>       sets build-directory to <dir> for the fli4l-files
-v, --verbose           verbose - some debug-output
    --filesonly         creates only fli4l-files - does not create a disk
    --no-squeeze        don't compress shell scripts
-h, --help              display this usage
```

```
config-dir              sets other config-directory - default is "config"

*** Remote-Update options
--remoteupdate          remote-update via scp, implies "--filesonly"
--remoteuser <name>     user name for remote-update - default is "fli4l"
--remotehost <host>     hostname or IP of remote machine - default
                        is HOSTNAME set in [config-dir]/base.txt
--remotepath <path>     pathname on remote machine - default is "/boot"
--remoteport <portnr>   portnumber of the sshd on remote machine

*** GUI-Options
--nogui                 disable the config-GUI
--lang                  change language
                        [deutsch|english|espanol|french|magyar|nederlands]
```

An alternative configuration directory can be passed by appending its name to the end of the command line. The normal configuration directory is called `config` and can be found under the fli4l root directory. This is where all fli4l packages place their configuration files. If you want to maintain more than one configuration, create another directory, e.g. `hd.conf`, place a copy of the configuration files there and change it according to the requirements. Here are some examples:

```
mkfli4l.bat hd.conf
mkfli4l.bat -v
mkfli4l.bat --no-gui config.hd
```

5.2.2. Configuration dialog – Setting the configuration directory

In the main window the configuration directory setting is indicated and a window can be opened for the selection of the configuration directory.

It should be noted that any change in the 'Config-Dir' causes all options to be set to the values contained in the control file 'mkfli4l.txt' (Page **??**) placed in that directory, or to the values given as command-line parameters, respectively.

If mkfli4l.bat does not find a directory fli4l-x.y.z\config or if there is no file in that directory named 'base.txt', a window is immediately opened for the selection of the configuration directory. This makes it possible to easily manage several fli4l configuration directories in a simple manner.

Example:

```
fli4l-x.y.z\config
fli4l-x.y.z\config.fd
```

```
fli4l-x.y.z\config.cd
fli4l-x.y.z\config.hd
fli4l-x.y.z\config.hd-create
```

5.2.3. Configuration dialog – General Preferences

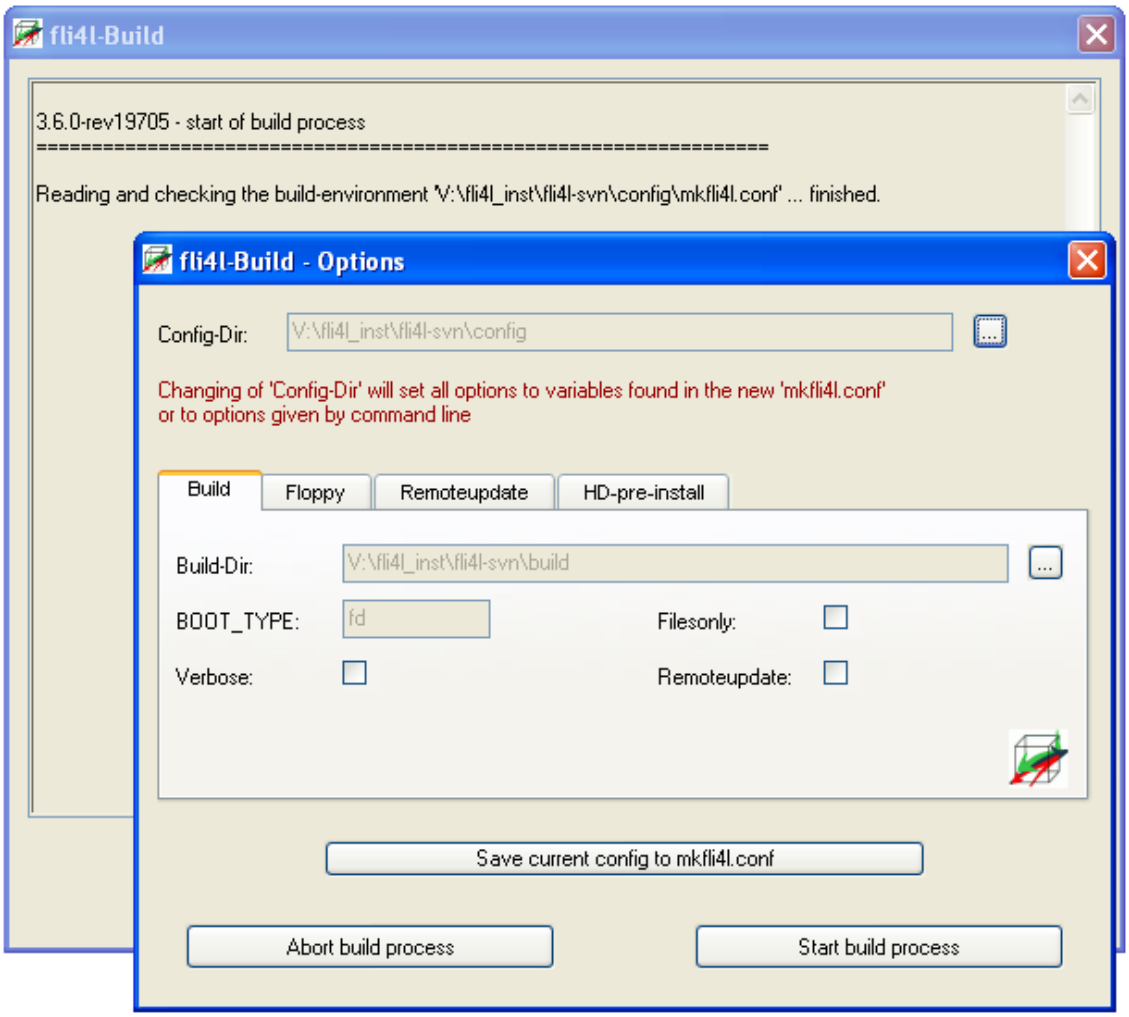


Figure 5.1.: Preferences

In this dialogue the settings are specified for the archive/boot-media creation:

- Build-Dir – Directory for the Archives/CD-Images/...
- `BOOT_TYPE` – Display of the utilized/settings `BOOT_TYPE` – unchangeable
- Verbose – Activation of additional output during the creation
- Filesonly – Only the archives are created – no bootmedia/no image
- Remoteupdate – Activation of the remote update via SCP

Using the button **Current settings in mkfli4l.txt buffer** the current settings can be stored in mkfli4l.txt.

5.2.4. Configuration dialog – Settings for Remote update

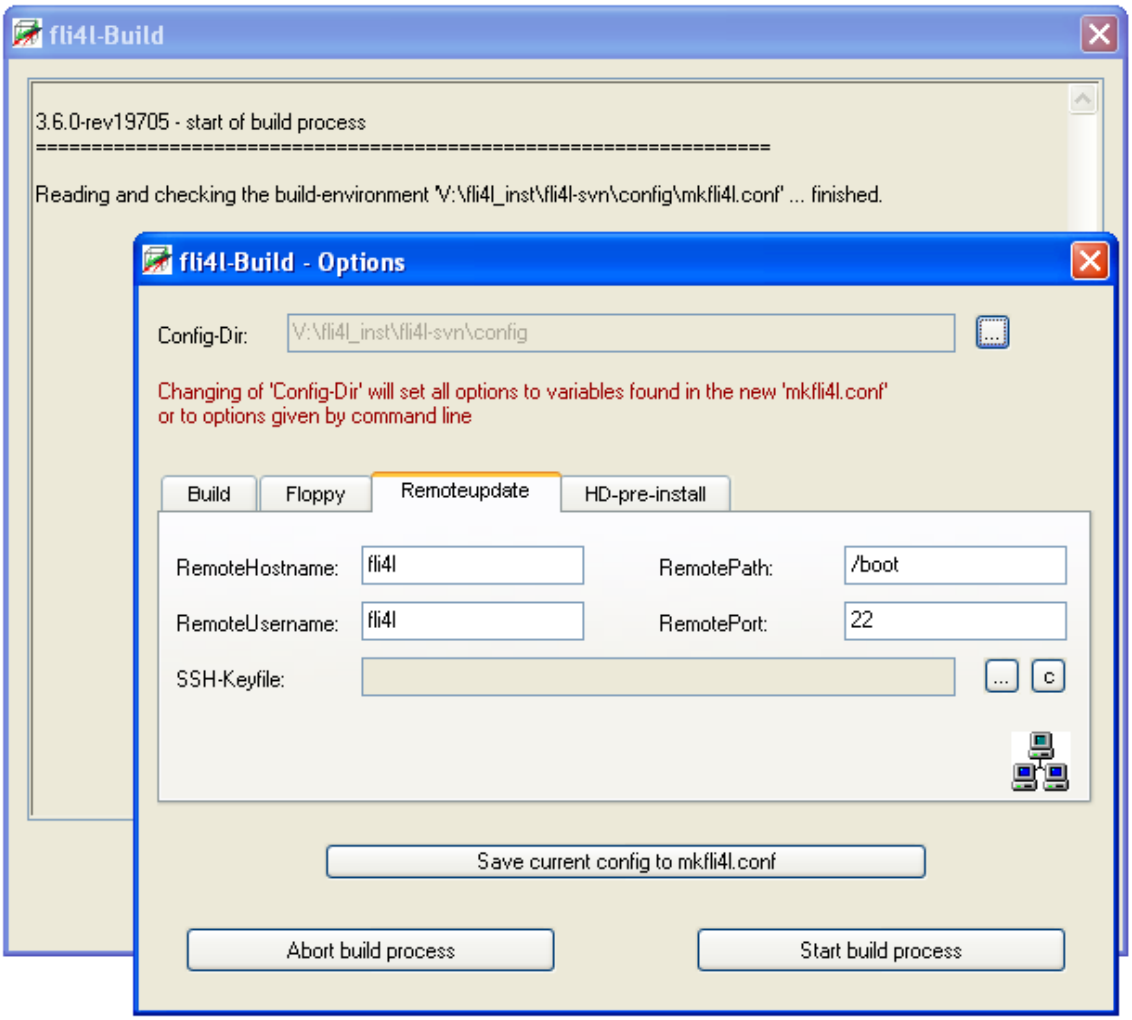


Figure 5.2.: Settings for Remote update

In this dialogue the settings for Remote update are specified:

- IP address or Hostname
- User name on the Remote host
- Remote path (default: /boot)
- Remote port (default: 22)
- SSH keyfile to use (format ppk from Putty)

5.2.5. Configuration dialog – Settings for HD pre-install

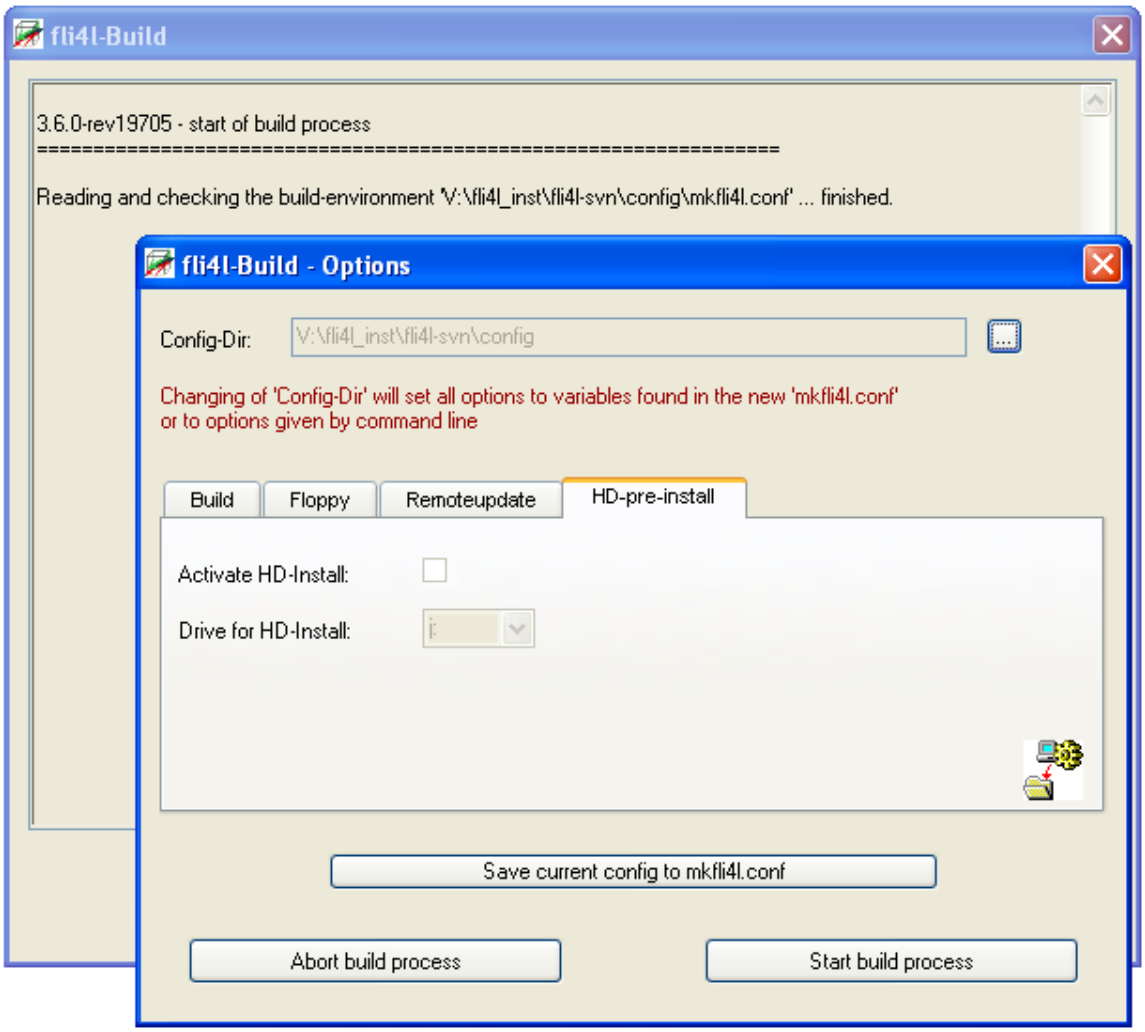


Figure 5.3.: Settings for HD pre-install

In this dialogue the options are set for HD pre-install on an accordingly partitioned and formatted Compact Flash card in a USB reader.

Possible Options:

- Activate HD pre-install
- Drive letter to be used to access the CF card

Regarding the partitioning and formatting of the CF: A Type-A HD installation (see package HD) must be based on a primary, active, and formatted FAT partition on the CF card. If you would like to use a data partition additionally, a Linux partition which is formatted with the ext3 file system, as well as the file `hd.cfg` are also needed on the FAT Partiton (in this case please make sure to read the documentation of the HD package).

5.3. Control file mkfli4l.txt

Since fli4l-Version 2.1.9 the control file <config>/mkfli4l.txt exists. This file can e.g. be used to specify directories which differ from the standard settings. The control file has a similar structure as the normal fli4l configuration files. All configuration variables here are optional, i.e. they need not exist or they can be commented out.

BUILDDIR Default: 'build'

Specifies the directory where fli4l files will be created. If the variable is undefined, the Windows mkfli4l sets it to 'build' relative to the fli4l root directory, resulting in the directory. build in the fli4l root directory:

```
Path/fli4l-x.y.z/build
```

Under *nix mkfli4l is using <config>/build and is thus filing the generated files together with the configuration.

The path for BUILDDIR must use the conventions of the Operating Systems Windows oder *nix. If relative paths configured there are converted by the build to the syntax of windows or *nix.

VERBOSE Default: VERBOSE='no'

Possible values are 'yes' or 'no'. Controls the *Verbosity* of the Build Processes.

FILESONLY Default: FILESONLY='no'

Possible values are 'yes' or 'no'. This will actually turn off the creation of the boot-media and only the files will be created –

REMOTEUPDATE Default: REMOTEUPDATE='no'

Possible values are 'yes' or 'no'. Enables automatic transferring of files by means of SCP to the router. This requires the package SSHD (Page 217) with activated scp. See also the following variables.

REMOTEHOSTNAME Default: REMOTEHOSTNAME="

The target host name for the SCP data transfer. If no name is set, the variable HOSTNAME (Page 24) is used.

REMOTEUSERNAME Default: REMOTEUSERNAME='fli4l'

User name for the SCP data transfer.

REMOTEPATHNAME Default: REMOTEPATHNAME='/boot'

Destination path for the SCP data transfer.

REMOTEPORT Default: REMOTEPORT='22'

Destination port for the SCP data transfer.

SSHKEYFILE Default: SSHKEYFILE="

Here you can specify a SSH key file for the SCP Remote update. Thus, an update can be made without specifying a password.

REMOTEREMOUNT Default: `REMOTEREMOUNT='no'`

Possible values are `'yes'` or `'no'`. If `'yes'` is set, a boot device "/boot" mounted read-only will be remounted read-write to allow remote updates of the boot files.

TFTPBOOTPATH Path where the remote Netboot image is saved to.

TFTPBOOTIMAGE Name of the Netboot image.

PXESUBDIR Subdirectory for the PXE files relative to TFTPBOOTPATH.

SQUEEZE_SCRIPTS Enable or disable the Squeezing (Compression) scripts. Compressing a script with Squeeze removes all comments and line indentations. Under normal conditions the default value of `'yes'` can be used.

MKFLI4L_DEBUG_OPTION Additional debugging options can be handed over to themkfli4l-Programm (Page 283).

6. Connecting PCs in the LAN

For every host in the LAN you will have to set up:

1. IP address (see IP address)
2. Name of the host plus desired domain name (see Host and domain name)
3. Default gateway (see Gateway)
4. IP address of the DNS server (see DNS server)

6.1. IP address

The IP address of the host has to belong to the same network as the IP address of the fli4l router (on the Ethernet interface), for example 192.168.6.2 in case the router has the IP address 192.168.6.1. IP addresses have to be unique throughout the network, so it's a good idea to change (only) the last number. You will also have to make sure you specify the same IP address as specified in the file config/base.txt.

6.2. Host and domain name

The name of the host is for example "my-pc", the domain "lan.fli4l".

Important: *The domain set up on the host has to be identical to the domain set up on the fli4l if you want to use fli4l as a DNS server. Otherwise it could cause massive problems in the network.*

The reason: Windows hosts regularly search for hosts within their workgroup, trying to resolve the name WORKGROUP.my-domain.fli4l. If the domain (here: my-domain.fli4l) doesn't match the one set up on the router, fli4l will try to answer the query by forwarding it to the Internet ...

The domain has to be entered in the TCP/IP settings of the host.

6.2.1. Windows 2000

On Windows 2000 the settings can be found under:
Start ⇒
Settings ⇒
Control Center ⇒
Network and Dial-up Connections ⇒
LAN Connection ⇒
Properties ⇒
Internet protocol (TCP/IP) ⇒

Properties ⇒
Extended... ⇒
DNS ⇒
Add DNS-Suffix ⇒

Type “lan.fli4l” (or the domain set up – without “”!) ⇒Click OK.

6.2.2. NT 4.0

Start ⇒
Settings ⇒
Control Center ⇒
Network ⇒
Protocols ⇒
TCP/IP ⇒
Properties ⇒
DNS ⇒

- Enter hostname (of the client)
- Enter domain (same as in config/base.txt)
- Add IP address of fli4l router
- Add DNS suffix (add domain – see two lines above)

6.2.3. Win95/98

Start ⇒
Settings ⇒
Control Center ⇒
Network ⇒
Configuration ⇒
TCP/IP (the one that is bound to the network
interface to the router) ⇒
Properties ⇒
DNS Configuration:
Activate DNS and under “Domain:” enter “lan.fli4l” (or the domain set up – without “”!)

6.2.4. Windows XP

On Windows XP the settings can be found at:
Start ⇒
Settings ⇒
System Settings ⇒
Network Connections ⇒
LAN-Connection⇒
Properties ⇒

Internetprotocol (TCP/IP) ⇒
Properties ⇒
Advanced. . . ⇒
DNS ⇒
DNS-Suffix for this connection ⇒

Specify “lan.fli4l” (resp. the domain you use) (without “”!) ⇒Press OK.

6.2.5. Windows 7

On Windows 7 the settings can be found at:
Windows Button (ex. Start) ⇒
System settings ⇒
Network and Internet ⇒
Network- and Sharecenter ⇒
LAN-Connection⇒
Properties ⇒
Internetprotocol Version 4 (TCP/IPv4) ⇒
Properties ⇒
Advanced . . . ⇒
DNS ⇒
DNS-Suffix for this connection ⇒

Specify “lan.fli4l” (resp. the domain you use) (without “”!) ⇒Press OK.

6.2.6. Windows 8

On Windows 8 the settings can be found at:
Press Windows- and X-key simultaneously ⇒
System settings ⇒
Network and Internet ⇒
Network- and Sharecenter ⇒
Choose your net (Ehternet or WLAN) ⇒
Properties ⇒
Internetprotocol Version 4 (TCP/IPv4) ⇒
Properties ⇒
Advanced . . . ⇒
DNS ⇒
DNS-Suffix for this connection ⇒

“lan.fli4l” (bzw. die eingestellte domain) eingeben (ohne “”!) ⇒OK drücken.

6.3. Gateway

It is absolutely necessary to specify a default gateway, because without the correct IP address provided here nothing will work. So you will have to specify the IP address of the fli4l router

here (the Ethernet interface's one) – for example 192.168.6.4, depending on the IP address that has been specified in the file config/base.txt for the router.

It is wrong to enter fli4l as a proxy in the Windows or browser configuration unless you use a proxy on the router. Normally fli4l is not a proxy, thus please do *not* specify fli4l as a proxy!

6.4. DNS server

As for the IP address, you should not specify the IP address of the provider's DNS server, but the address of the router (Ethernet interface), as it will anwser queries itself or forward them to the Internet if needed.

When fli4l is used as a DNS server, many queries from Windows hosts are not forwarded to the Internet but answered by fli4l itself.

6.5. Miscellaneous

The items 1 to 4 do not have to be specified when a DHCP server is configured as fli4l transmits the needed data automatically.

Internet options: Under connections you will have to select "do not dial". Under settings for local network (LAN): Do NOT enter anything (unless you use `OPT_Proxy`). Both are default settings and should already exist.

7. Client/Server interface imond

7.1. imon-Server imond

imond is a network-capable server program that responds to certain queries or accepts commands that can control the router.

imond also controls the Least-Cost-Routing. It uses the configuration file /etc/imond.conf, that is created automatically from the variables `ISDN_CIRC_x_XXX` from the file config/isdn.txt and other at boot time by a shell script.

imond runs permanentely as daemon and listens on TCP/IP port 5000 and the device /dev/isdninfo.

All possible commands that can be sent to TCP/IP port 5000:

The TCP/IP port 5000 is only reachable from the masqueraded LAN. Access from remote is blocked by the firewall configuration by default.

Imond supports two user levels: the user and the admin mode. For both levels you can set a password using varIMOND_PASS and/or `IMOND_ADMIN_PASS`. Then, clients are forced by imond to submit a password. As long as no password has been submitted, only the commands "pass" and"quit" are accepted. Others are rejected.

If you want to further restrict access, e.g. only allow access from a single computer, the firewall configuration has to be changed.

At present this is not possible using the standard configuration files config/base.txt. You will have to change the file /etc/rc.d/rc322.masq.

The commands

```
enable/disable/dialmode   dial/hangup   route   reboot/halt
```

Can be globally enabled/disabled using the configuration variables `IMOND_XXX` (see "Configuration").

From a Unix/Linux computer (or a Windows computer in a DOS box) you can easily try it out: Type

```
telnet fli4l 5000        \# or the appropriate name of the fli4l-Routers
```

and you will be able to directly enter the listed commands and look at the output.

For example after entering "help" the help is shown, after "quit" the connection to imond is terminated.

7.1.1. Least-Cost-Routing – how it works

imond contructs a table (time table) from the configuration file /etc/imond.conf (which is created on bootup from the config variables `ISDN_CIRC_x_TIMES` and others). It contains a complete calendar week in a raster of 1 hour (168 hours = 168 Bytes). But the table only contains the circuits that have a default route defined.

Admin commands

addlink ci-index	Add channel to the circuit (channel bundling)
adjust-time seconds	Increments the date on the router by the number of seconds specified
delete filename pw	Deletes the file on the router
hup-timeout #ci-index [value]	Show or set the HUP timeout for ISDN circuits
removelink ci-index	Remove additional channel
reset-telmond-log-file	Deletes the telmond log file
reset-imond-log-file	Deletes the imond log file
receive filename #bytes pw	Transfer a file to the router. Imond acknowledges the command using an ACK (0x06). After that, the file is transfered in blocks of 1024 bytes that are also acknowledged with an ACK. Finally, imond replies with an OK.
send filename pw	If the password is correct and the file exists, imond replies with OK #bytes. Then, imond transfers the file in blocks of 1024 bytes that have to be acknowledged with an ACK (0x06). Finally, imond replies with an OK.
support pw	Shows the status/configuration of the router
sync	Syncs the cache of mounted drives

Admin or User commands

dial	Dials the provider (Default-Route-Circuit)
dialmode [auto|manual|off]	Shows or sets the dialmode
disable	Hangs up and sets the dialmode to “off”
enable	Sets the dialmode to “auto”
halt	Cleanly shuts down the router
hangup [#channel-id]	Hangs up
poweroff	Shuts down the router and powers it off
reboot	Reboots the router
route [ci-index]	Set the default route to circuit X (0=automatically)

User commands

channels	Shows the number of available ISDN channels
charge #channel-id	Shows the online fee for a specific channel
chargetime #channel-id	Time charged in consideration of the charge interval
circuit [ci-index]	Shows a circuit name
circuits	Shows number of default-route-circuits
cpu	Shows the CPU load in percent
date	Shows date/time
device ci-index	Shows the device of the circuit
driverid #channel-id	Shows driver-id of the channel X
help	Shows help
inout #channel-id	Shows direction (incoming/outgoing)
imond-log-file	Shows imond log file
ip #channel-id	Shows IP
is-allowed command	Shows, whether a command is configured/allowed Possible commands: dial|dialmode|route|reboot|imond-log|telmond-log|mgetty-log
is-enabled	Shows, whether dialmode is off (0) or auto (1)
links ci-index	Show number of channels 0, 1 or 2, 0 means: No channel bundling possible
log-dir imond|telmond|mgetty	Shows the log directory
mgetty-log-file	Shows mgetty logfile
online-time #channel-id	Shows online time of the current connection in hh:mm:ss
pass [password]	Show, whether it is necessary to enter a password or enter a password 1 Userpassword is set 2 Adminpassword is set 4 imond is in admin mode
phone #channel-id	Show telephone number/name of the peer
pppoe	Show the number of pppoe devices (i.e. 0 or 1)
quantity #channel-id	Show the data transferred in bytes
quit	Terminates the connection to imond
rate #channel-id	Show transfer rates (incoming/outgoing in B/sec)
status #channel-id	Show status of channel X
telmond-log-file	Shows telmond log files
time #channel-id	Show the sum of online times, format hh:mm:ss
timetable [ci-index]	Shows the time table for the LC-Routing
uptime	Shows the uptime of the router in seconds
usage #channel-id	Show the type of connection, that is available responses: Fax, Voice, Net, Modem, Raw
version	Show the protocol and program version

Using the imond command “timetable” you can have a look at it.
Here an example:
Supposing 3 circuits are defined:

```
CIRCUIT_1_NAME='Addcom'
CIRCUIT_2_NAME='AOL'
CIRCUIT_3_NAME='Firma'
```

Only the first two circuits have a default circuit defined, i.e. the corresponding variables ISDN_CIRC_x_ROUTE have the value ’0.0.0.0’.
If the variables `ISDN_CIRC_x_TIMES` look like this:

```
ISDN_CIRC_1_TIMES='Mo-Fr:09-18:0.0388:N Mo-Fr:18-09:0.0248:Y
              Sa-Su:00-24:0.0248:Y'

ISDN_CIRC_2_TIMES='Mo-Fr:09-18:0.019:Y Mo-Fr:18-09:0.049:N
              Sa-Su:09-18:0.019:N Sa-Su:18-09:0.049:N'

ISDN_CIRC_3_TIMES='Mo-Fr:09-18:0.08:N Mo-Fr:18-09:0.03:N
              Sa-Su:00-24:0.03:N'
```

it results in the following /etc/imond.conf being created:

```
#day   hour   device  defroute  phone        name        charge  ch-int
Mo-Fr 09-18 ippp0   no        010280192306 Addcom      0.0388   60
Mo-Fr 18-09 ippp0   yes       010280192306 Addcom      0.0248   60
Sa-Su 00-24 ippp0   yes       010280192306 Addcom      0.0248   60
Mo-Fr 09-18 ippp1   yes       019160       AOL  0.019   180
Mo-Fr 18-09 ippp1   no        019160       AOL  0.049   180
Sa-Su 09-18 ippp1   no        019160       AOL  0.019   180
Sa-Su 18-09 ippp1   no        019160       AOL  0.049   180
Mo-Fr 09-18 isdn2   no        0221xxxxxxx  Firma       0.08     90
Mo-Fr 18-09 isdn2   no        0221xxxxxxx  Firma       0.03     90
Sa-Su 00-24 isdn2   no        0221xxxxxxx  Firma       0.03     90
```

imond creates the following time table in memory – here the output of the imond command “timetable”:

```
    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
-------------------------------------------------------------------------
Su  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
Mo  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4  4  4  2  2  2  2  2  2
Tu  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4  4  4  2  2  2  2  2  2
We  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4  4  4  2  2  2  2  2  2
Th  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4  4  4  2  2  2  2  2  2
Fr  2  2  2  2  2  2  2  2  2  4  4  4  4  4  4  4  4  4  2  2  2  2  2  2
Sa  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3

No.  Name                 DefRoute  Device  Ch/Min   ChInt
 1   Addcom                 no      ippp0   0.0388     60
 2   Addcom                 yes     ippp0   0.0248     60
 3   Addcom                 yes     ippp0   0.0248     60
```

```
 4   AOL              yes     ippp1   0.0190    180
 5   AOL              no      ippp1   0.0490    180
 6   AOL              no      ippp1   0.0190    180
 7   AOL              no      ippp1   0.0490    180
 8   Firma                    no      isdn2   0.0800     90
 9   Firma                    no      isdn2   0.0300     90
10   Firma                    no      isdn2   0.0300     90
```

For circuit 1 (Addcom) there are three time ranges (1-3) defined. For circuit 2 (AOL) there are four time ranges (4-7) and for the last one there are three time ranges (8-10).

In the time table, the indices are printed that are valid in the corresponding hour. Only the indices 2-4 show up here, as the others are not default routes.

If there are zeros in the table, there are gaps in the values of the `ISDN_CIRC_X_TIMES` variables. At this point there is no default route, no internet access is possible!

On program start, imond checks for the weekday and the hour. Then, the index from the time table is picked out and the corresponding circuit. The default route is then set to this circuit.

If the status of a channel changes (e.g. offline – online) or at least after one minute, this procedure is repeated: check time, lookup in table, pick default route circuit.

If the used circuit changes, e.g. on mondays, 18:00, the default route is deleted, existing connections are terminated (sorry...) and after that the default route is set to the new circuit. Imond may notice this up to 60 seconds too late – so at least at 18:00:59 the route is changed.

If a circuit does not have a default route, nothing will change. The value of `ISDN_CIRC_x_TIMES` is only used to calculate the fee. This can be important if the LC routing is disabled temporarily, e.g. using the client imonc, and a circuit is dialed manually.

But you can have a look at the tables for other time-range-indices (in our example 1 to 10), also at the ones of the "Non-LC-Default-Route-Circuits".

Command:

```
timetable index
```

Example:

```
telnet fli4l 5000
timetable 5
quit
```

The output will look like:

```
    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
-------------------------------------------------------------------------
Su  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Mo  5  5  5  5  5  5  5  5  5  0  0  0  0  0  0  0  0  0  5  5  5  5  5  5
Tu  5  5  5  5  5  5  5  5  5  0  0  0  0  0  0  0  0  0  5  5  5  5  5  5
We  5  5  5  5  5  5  5  5  5  0  0  0  0  0  0  0  0  0  5  5  5  5  5  5
Th  5  5  5  5  5  5  5  5  5  0  0  0  0  0  0  0  0  0  5  5  5  5  5  5
Fr  5  5  5  5  5  5  5  5  5  0  0  0  0  0  0  0  0  0  5  5  5  5  5  5
Sa  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

No.  Name                 DefRoute  Device  Ch/Min   ChInt
 5   AOL                     no      ippp1   0.0490    180
```

Got everything?

Using the command “route”, the LC routing can be enabled or disabled. If a positive circuit index is specified (1...N) the default route is changed to the circuit specified. If the index is 0, LC routing will be activated again and the active circuit is chosen automatically.

7.1.2. Annotations to the calculation of the online changes

The whole model how the online charges are calculated will only work correctly, if the charge interval for a single circuit (variable `ISDN_CIRC_x_CHARGEINT`) remains constant throughout the whole week.

Normally, this is correct for most of the internet providers. But if you dial in, e.g. to your companies network, using the (German) Telecom (not the internet provider T-Online), the change interval changes at 18:00 from 90 seconds to 4 minutes (information from June 2000). Because of that, the definition

```
ISDN_CIRC_3_CHARGEINT='90'
ISDN_CIRC_3_TIMES='Mo-Fr:09-18:0.08:N Mo-Fr:18-09:0.03:N Sa-Su:00-24:0.03:N'
```

is not absolutely correct. After 18:00 the price is converted to 3 cents (4 minutes cost 12 cents), but the charge interval is wrong. Because of that, the displayed charge could differ from the actual one.

Here is a tip, how different charge intervals can be handled correctly, anyhow (also important for `ISDN_CIRC_x_CHARGEINT`): Just define 2 cicuits – one for each charge interval. Of course you will have to adjust `ISDN_CIRC_x_TIMES` so that the valid circuit is always dialed, depending on the charge interval.

Once again: If you connect to an ISP you most likely will not have this problem, because the charge interval is constant all the time and only the prices per minute change (or doesn't it? I guess the German provider T-* could even introduce such a price model :-).

7.2. Windows-Client imonc.exe

7.2.1. Introduction

imond on the router and on the client imonc as a team have two use modes: the user and the admin mode. In admin mode, all controls are enabled. In user mode the variables `IMOND_ENABLE` (Page 70), `IMOND_DIAL` (Page 69), `IMOND_ROUTE` (Page 69) and `IMOND_REBOOT` (Page 69) control if the respective functions are available. If all of these variables are set to ‘no ’, this means that all buttons in the overview page are disabled except for the exit and the admin mode button. The decision whether the user or admin mode is used, is based on the transmitted password. By clicking the button admin mode, located in the status bar, you may switch to the admin mode at any time by entering the admin password. To switch back imonc must be restarted.

Once imonc is started an additional tray icon is displayed, which shows the connection status of existing channels.

The colors mean:

Red : Offline

Yellow : A connection is in the process of being established

Light Green : Online and traffic on the channel

Dark Green : Online and (nearly) no traffic on the channel

imonc shows a behavior a little different from the Windows standard when clicking on the minimize button in the title bar. This minimizes imonc to the system tray only a tray icon near the clock remains. Double clicking on the tray icon with the left mouse button brings imonc's window back to the foreground. With the right mouse button you may use the context menu, of the tray icon to execute the main imonc commands die angezeigten without displaying its main window.

Imonc stores many properties (including all columns widths of the string grids) in the registry, so its appearance may be widely adapted to your needs. Imonc uses the registry key HKCU\Software\fli4l to store this informations

If even after careful reading of the documentation problems on imonc or the router itself still remain you can post to the newsgroup. It makes sense to note the support information you get when choosing SystemInfo and then Info on the About page of the imonc client. The router password will be queried then (not the imond password) and after that imonc will create a file fli4lsup.txt that contains all the important information regarding the router, including imonc. This file can be posted on the newsgroup when explicitely demanded and will give much better chance for quick help.

Further details concerning the development of the Windows imonc client can be found on the homepage of Windows Imonc http://www.imonc.de/. Here you can see what new features and bug fixes will be included in the next version. In addition, you will find the latest imonc version, if it is not included in the FLI4L distribution already.

7.2.2. Start Parameters

Imonc requires the name or IP address of the router fli4l. As the default the program attempts to establish a connection with the computer "fli4l". If this entry is correctly entered in the DNS, this should work out of the box. Otherwise you can pass following parameters in the link:

- /Server:The router's IP or hostname (short form: /S:IP or hostname)
- /Password:password (short form: /P:password)
- /log Option for logging communication betweeen imonc und imond. When entered a file imonc.log will be written each time when imonc exits. It contains the complete communication and thus can grow quite large. Please use this parameter only in case of problems.
- /iport:Portnumber Portnumber imond listens to. Default: 5000
- /tport:Portnummer Portnumber telmond listens to. Default: 5001
- /rc:"Command" The command provided here will be transmitted to the router without further checking and imonc will exit afterwards. If more than one command should be transmitted at once, they must be devided by semicolons. You will have to provide an imond password in addition for this to work because no password query will be queried. Possible command are documented with imond, see chapter 8.1. In addition to

the commands there another one exists: timesync. If used imonc will synchronize the clock of the client with the router's clock. The command dialtimesync is not supported anymore, it is substituted by "dial; timesync".

- /d:"fli4l-Directory" Pass the fli4l-directory as a start parameter. May be of interest when using more than one fli4l version.
- /wait If the hostname can't be reolved imonc will not exit anymore – start another try by doubleclicking the tray icon.
- /nostartcheck Disables checking of imonc already running. Only makes sense to monitor several different fli4l routers in a net. When using more instances the builtin syslog- and E-Mail-capabilites will be disabled.

Usage (to be entered in the link):

```
X:\...imonc.exe [/Server:Host] [/Password:Password] [/iport:Portnumber]
            [/log] [/tport:Portnumber] [/rc:"Command"]
```

Example with IP address:

```
C:\wintools\imonc /Server:192.168.6.4
```

or with name and password:

```
C:\wintools\imonc /S:fli4l /P:secret
```

or with name and password and router command:

```
C:\wintools\imonc /S:fli4l /P:secret /rc:"dialmode manual"
```

7.2.3. Overview

The Windows client queries some imond information on the existing connections and displays it in its window. In addition to general status information such as uptime of the router or the time (both locally and on the router), for each existing connection the following informations are shown:

Status	Connection establishment/Online/Offline
Name	Telephone number of the caller or circuit name
Direction	Indicates if a connection is incoming or outgoing
IP	IP, that was assigned to the router
IBytes	Bytes received
OBytes	Bytes sent
Online time	Actual online time
Time	Sum of all online times
KTime	Sum of all online times in consideration of charge intervals
Cost	Computed costs

The data is updated every 2 seconds by default. In the context menu of this overview it is possible to copy the assigned IP to the clipboard as well as hanging up the channel (for

each available channel which is online at the moment). This is of interest in case that several different connections exist, e.g. one to surf the Internet and another to a private net and only one of them should be terminated.

If the telmond process is active on the router, imonc can show information about incoming phone calls (ie calling and called MSN) in addition. The last incoming phone call is displayed above the buttons. a log of phone calls received can be obtained by viewing the calls page.

By the six buttons in imonc the following commands can be selected:

Button	Caption	Function
1	Connect/Disconnect	Dial/Hang up
2	Add link/Rem link	Bundle channels: yes/no – only available in admin mode
3	Reboot	Reboot fli4l!
4	PowerOff	Shut down fli4l and power off afterwards
5	Halt	Shut down fli4l to power off safe afterwards
6	Exit	Exit client

The first five commands can be switched on and off individually in the router's configuration file config/base.txt for the user mode. In admin mode all are enabled. The dialmode controls the dialing behavior of the router:

Auto	The router will establish connections automatically when getting queries from the local net for the circuit concerned.
Manual	The user himself has to establish connections.
Off	No connections can be established. The dial button is deactivated.

Note that fli4l by default will dial out independently. So you never actually will have to press the connect button...

It is also possible to manually switch the default route circuit, setting the automatic LCR on and off. In the Windows version of imonc the selection list " default route" is provided for this. In addition you can configure the hangup TimeOut time directly in imonc. use the Config button next to the default route for this. All configured circuits of the router are displayed here. The value in the column Hup-timeout can be edited directly in the string grid for ISDN circuits (not yet working for DSL).

An overview over LCR can be found on the page admin/Timetable. There you'll see what circuit imond selects automatically at which time.

7.2.4. Config-Dialog

The configuration is reached using the Config button in the status bar. The window is divided into the following areas:

- The Area General:
  - Actualization Interval: Set here how often the overview should get actualized.
  - Synchronize Time on Startup: When starting the client's system time and date will get synchronized with the router's system time. You can execute this manually by using the button Synchronize on the Overview-page.
  - Start Minimized: Program will start minimized to the system tray.

  - Start with Windows: Specify here if the client should start automatically with system start. Provide necessary start-parameters in the field Parameter.
  - Fetch News from fli4l.de: Should news from fli4l's homepage be fetched and displayed by imonc? Then headlines are shown in the status bar. A new page News is displayed to show the complete messages.
  - Logfile for Connections: The file name here is used to save connection lists locally on the imonc's system.
  - TimeOut for router to answer: How long should we wait for an answer from the router before assuming that the connection has failed.
  - Language: Pick the language for imoncs to use.
  - Confirm Router Commands: If activated all commands influencing the router generally have to be confirmed, i.e. Reboot, Hangup ...
  - Hang up even when traffic: No information should be displayed when the connection is closed and there is still traffic on the line.
  - Automatic Connection to the router: Should we try to reconnect to the router in case of lost connections (i.e. when rebooting the router).
  - Minimize Window To System Tray: Should imonc minimize to system tray instead of terminating itself when clicking the Exit button.

- Proxy Settings: Define a proxy for imonc's http-queries here. It will be used for fetching news.
  - Activate Proxy-support for Http-queries: Should we use a proxy
    * Address: Address of the proxy server
    * Port: Port number of the proxy server (default: 8080)

- TrayIcon: Set the colors of the tray icon next to the clock to your own needs. In addition you can specify that the actual dialmode will set the background color of the tray icon.

- Calls: The position of the call notification window will be saved in the registry in order to allow to set a fixed position for the window. Simlpy drag the window to the desired position.
  - Update: Set here in what way imonc will be informed about new calls. There are three possibilities. The first is querying the telmond service on the router in regular time intervals. A second is evaluating the syslog messages. This variant is preferred to the first - of course, the imon's syslog client has to be enabled. If imonc is used with a routed eisfair system the third possibility is to use the Capi2Text package for call signaling.
  - Delete Leading Zeros (Phone Boxes): Phone boxes often use an additional Zero to prefix the caller's number. This option will suppress the digit.
  - Own Area Code: Save your own area code here. If a call with the same area code is received it will get suppressed when displaying.
  - Telephone Book: Here, the file can be specified, in which the local Phone book is saved for resolving of the phone number. If the file does not exist, it is created by the program.

  - Logfile: The file name you can specify here is used to save the call list locally on the computer. This menu item is only visible if the config variable `TELMOND_LOG` is set to 'yes' (this also applies to the call list).
  - Use External Search: In this area, a program may be specified that will be called when a phone number can not be resolved using the local phone book. Info should be provided by the corresponding program. Until now there exists a connection to the telephone CD KlickTel and from Marcel Wappler a connection to the Palm database.

- Call Notification: Here can be determined whether an indication of incoming phone calls should be displayed, and how this is presented visually
  - Activate Call Notification: Indicate Calls or not.
  - Show Call Notification: Should a notification window be displayed on incoming calls? Infos: MSN called, Calling ID of the caller and date/time. Set variable `OPT_TELMOND` to 'yes' in config/isdn.txt for this to work.
    * Suppress If no number is transmitted: Should the call notification be displayed if no calling number was transferred?
    * Display Time: This setting specifies how long the call notification window should remain open. Setting this to "0" willavoid that the window is closed automatically.
    * Fontsize: Sets the font size. This is of influence for the window size because it is computed based on the font size.
    * Color: Set the font color here. I use red for better reecognition.

- Phonebook: The page Phonebook contains the numbers for resolving caller IDs (MSN). The page will be shown even if the variable `TELMOND_LOG` is set to 'no' caller number resolving is also used for showing the last aller on the summary page. Alternatively a local file can be picked as phone book here.

  The structure of the entry is as follows:

```
# Format:
# PhoneNumber=displayed Name[, Wavefilename]
# 0241123456789=Testuser
00=unknown
508402=Fax
0241606*=Elsa AG Aachen
```

  The first three lines are comments. The fourth line accomplishes that "unknown" will be shown if no caller ID is submitted. In the fifth line the name "Fax is assigned to MSN number 508402. In all other cases the format that will be shown is always PhoneNumber=Name. The sixth line demonstrates the possibility to define a group number. This will resolve all numbers matching the condition 0241606* to one name. Note that the first entry found in the phone book that matches will be picked. Optionally a wave-file can be set that will get played when a call with this number comes in.

As of version 1.5.2: on the page Names it is also possible to synchronize the local phone book with the router's one (stored in /etc/phonebook) and vice versa. The files are not simply replaced but missing entries will be added. If a phone number exists in both phone books with different name you will be prompted for the entry to be taken. Note that the synchronization of the phone book on the router is only changed in the ramdisk, so, after a reboot all changes will be lost.

- Sound: Wave-files specified here will be played when the specific event has occurred.
  - E-Mail: If E-Mail-Checking finds new E-Mails on the POP3-Server specified, the selected wave-file will be played.
  - E-Mail-Error: If an error occurs when loading E-Mails auftritt, this wave-file will be played.
  - Connection lost: When the connection to the roter is gone (i.i. the router is rebooted), this wave-file will be played. If the option "Automatic Reconnect to router" is not activated a MessageBox will pop up asking you to reconnect.
  - Connection Notification: When establishing a connection this wave-file will be played.
  - Connection closed: When a connection is closed this wave-file will be played.
  - Call Notification: If Call Notification is activated this wave-file will be played on incoming calls.
  - Fax Notification: If a new FAX is received this wave-file will be played.
- E-Mails
  - Accounts: Configure your POP3-Accounts here.
  - Activate E-Mail-Check: Should E-Mail-check look for new E-Mails automatically?
    - Check every x Min: How often should the E-Mail-check look for E-Mails automatically. Attention: a short interval can keep the router online forever! This will be the case if the interval is chosen shorter than the Hangup-Timeout of the circuit in use.
    - TimeOut x Sec: How long should we wait for the POP3-Server until it answers? The value "0" means no timeout is in effect.
    - Also if Router is offline: The router will perform a dialin to look for E-Mails. After checking all POP3-accounts the connection will be shut again. To use this feature the Dialmode has to set to 'auto'. Attention: If not using a flatrate additional costs will arise!
    - Circuit to use: Which circuit should be used for checking E-Mails?
    - Stay online afterwards: Should the connection stay until Hangup-timeout or hung up directly after E-Mail-Check.
    - Load E-Mail-Header: Should the E-Mail-Headers be loaded instead of only queriying the number of E-Mails? Loading E-Mail-Headers is a precondition for deleting E-Mails directly on the server.
    - Notify only of new E-Mails: Should only be noted for new E-Mails acoustically and with the tray icon?

    * Start E-Mail-Client: Should the E-Mail-Client bes tarted automatically if new E-Mails were found?
    * E-Mail-Client: Specify the E-Mail-Client to start.
    * Param: Provide additional parameters for starting the E-Mail-Client. If using Outlook as E-Mail-Client (not Outlook Express), you should set /recyle as a parameter. This will use an already existing instance of Outlook when loading new E-Mails.

- Admin
  - root-Password: Set the router password (`PASSWORD` in config/base.txt) here, i.e. to edit port forwarding locally and copy it back to the router.
  - Files on the router that should be displayed: All router files mentioned here can be displayed on the page admin/files easily via a mouse click. This way you can review logfiles of the routers very easy directly in imonc.
  - Edit Config files: Choose here if config files should all be opened in an editor (if the TXT-files are linked to an external editor this may lead to a huge number of open editor instances). Alternatively only the directory can be opend to give you a chance to pick the files to rework yourself.
  - DynEisfaiLog: If an account at DynEisfair exists you may set the login data here to review a logfile for the actulization of the service on the page Admin/DynEisfairLog.
- LaunchList serves for configuring the launch list (did you guess?). If will be executed after a successful connect if the option "Activate Launchlist" is activated.
  - Programs: All programs mentioned here will be started automatically when the router established a connection and the launch list is activated.
  - Activate LaunchList: Should it be executed on a successful connect?
- Traffic serves for adapting the look of the TrafficInfo window to you needs. A user reported problems with older versiions of DirectX.
  - Separate Traffic-Info-Window: Should a graphical channel visualization be displayed in a separate window? In the context menu of the window you can define whether the window get the StayOnTop attribute. This causes the window to be placed on top of all other windows. This value is also saved in the registry and thus is available even after a program restart.
  - Show title bar: should the title bar of the traffic info window be displayed? It shows with which Circuit the router is online at the moment.
    * CPU usage in title bar: Should the CPU utilization be displayed in the title bar?
    * Online time in title bar: Should the online time of the channel also be displayed in the title bar?
  - Semi-transparent window: Should the window be transparent? This feature is available only on Windows 2000 and above.

  - Colors: Define the main colors for the Traffic Information window. It should be taken into account that the DSL channel and the first ISDN channel will be assigned the same color value.
  - Limits: Set the maximum transfer values for DSL here - upload and download.

- The syslog area is used to configure the display of syslog messages.
  - Activate Syslog-Client: Should imonc display syslog messages? This option be switched off if an external syslog client (for example Kiwi's Syslog Client) is used.
  - Show All Messages From: Messages should be shown from which priority on? It makes sense to start with debug priority to see which messages are interesting for you. After that you may set the priority to your preferance.
  - Save Syslog Messages To A File: Should syslog messages be saved to a file in addition? Choose the messages to be logged to the file in the groupbox. The following placeholders are present:

    %y – will be replaced by the current year

    %m – will be replaced by the current month

    %d – will be replaced by the current day
  - Show Port Names: Should we display port names instead numbers?
  - Firewall-Messages In User Mode: Specify here whether Firewall Messages should also be shown in user mode or not.

- The Fax Area serves to configure Fax display in imonc. This area only appears if mgetty resp. faxrcv are installed on the router (OPT- packages on fli4l's homepage).
  - Fax Logfile: The filename set here is used to save Fax lists locally.
  - Local Directory: To display Faxes they have to be stored locally. Set the directory destination for this option here.
  - Actualization: There are two different ways for imonc to recognize a new Fax that has been received. Either imonc monitors the syslog messages (the syslog-client in imonc must be activated then) or it checks the lofiles in intervals. Prefer the first option if possible. If using the second option you may specify the time interval to actualize the page Fax overview. Note that this setting is not given in seconds but will be multiplied with the setting in Common/Actualization interval.

- The area grids serves for adapting the tables in imonc to your own needs. Set for each grid which colums should be shown and for grids in the area calls, connections and Faxes since what time the infos should be displayed.

7.2.5. Calls Page

The calls page is only dsiplayed if the configuration variable `TELMOND_LOG` is set to 'yes' because no call log exists otherwise. All incoming calls that were logged while the router was working are displayed on this page. You may choose between viewing calls saved locally or on the router. When clicking on the reset button while reviewing the calls saved on the router the logfile there will be erased.

In the call overview you may right click on the number or MSN to copy it to the phone book and assign a name to it there which will shown instead from this point on.

7.2.6. Connections Page

As of version 1.4 this page displays the connections established by the router. This helps to monitor the router's behavior especially when automatic dialin is configured. `IMOND_LOG` in config/base.txt has to be set to 'yes' for this page to appear.

You may choose between viewing connections saved locally or on the router. When clicking on the reset button while reviewing the router's connection log it will be erased.

The following infos will be shown

- Provider
- Start date and -time
- End date und -time
- Online time
- Charged time
- Costs
- Inbytes
- Outbytes

7.2.7. Fax Page

Either `OPT_MGETTY` or `OPT_MGETTY` has to be installed on the router. You will find both on the fli4l homepage in the opt database. All incoming faxes will be listed on this page then. The context menu of the overview provides several options only available in admin mode:

- A Fax may be displayed. In Admin/Remoteupdate the fli4l directory path has to be set correctly because Faxes on the router are gzip-packed and thus need this program to exist in the path. You may also copy gzip.exe and win32gnu.dll to the imonc directory. If gzip.exe is not found at this places imonc tries to open the webserver of the router on the right page.
- A may be deleted. If chosen the Fax will be deleted locally and on the router (the fax file and the corresponding entry in the logfile).
- All Faxes o the router may be deleted. Files and logfile on the router are both deleted, but not from the local logfile.

You may switch between Fax overview local and on the router.

7.2.8. E-Mail Page

This page is shown only if at least one POP3-E-Mail-account is configured and activated in the config dialog.

The page E-Mail should be self-explaining. Here the E-Mail- Check is monitored. If the option "Check even if the router is offline" is not activated the E-Mail-Check will check all E-Mail-accounts for E-Mails in the specified tme interval when the router is online. If the option is activated the E-Mail-Check will go online if necessary with the circuit in use at this moment and after this close the connection again. Dialmode has to be set to "auto" for this to work.

If E-Mails are found on the POP3 server vorhanden either the configured E-Mail-Client will be started or a new symbol is shown in the try icon containing the number of E-Mails on the server. A double click will start the E-Mail-Client then. If an error occurred with one of the E-Mail-accounts a message is shown in the E-Mail-History and the E-Mail-TrayIcon shows a red colored upper right edge.

In the E-Mail-overview you may delete mails directly on the server by using the context menu (right click) without having to download them before. The cell of the E-Mail to be deleted should be selected before. Choose Delete MailMessage to perform the action.

7.2.9. Admin

This area is only visible if imonc is in admin mode.

The first item shows an overview on the circuits – resp. Internet providers – which the router can choose automatically via LCR. A double click on a provider will show the times defined for it in config/base.txt.

The second item enables you to do a remote update for the router. You may choose which from the five packages (Kernel, Root filesystem, Opt-file, rc.cfg and syslinux.cfg) should be copied to the router. To copy the update you have to specify the fli4l directory to inform imonc from where it should obtain the files needed. Also the subdirectory for the config files (default config) is needed for creating the Opt-file, rc.cfg and syslinux.cfg. A reboot should be preformed after the update to enable the new configuration. If a password is queried while updating the one from config/base.txt at PASSWORD is meant here.

To avoid port forwarding ony binding to exactly one client PC you may now edit the configuration directly on the router. For the change to come to effect you have to reconnect. Because the file is only edited in the ram disk all changes are lost with the next router reboot. To save your changes permanently you have to have to adapt the base.txt in config and update the Opt-File on the router.

The fourth item on the admin page – Files – is used for easy review of configuration and log-files simply via a mouse click. The list is configured in Config/Admin and then "files on router to view". After that you may pick which file to show in the ComboBox on this page.

The fifth item is the page DynEisfairLog. It only appears if you provided the access data for your DynEisfair account in the Config-file. The logs of the service will be displayed then.

The last item is the Hosts page. All computers in /etc/hosts are shown here. All these will be pinged and the result is shown as well. In this way you can check if a PC is on.

7.2.10. Error, Syslog and Firewall Pages

Those pages are only visible if entries are present in the respective logs and imonc is in admin mode.

An the errors page all imonc/imond-specific errors are noted. If problems occur reviewing this page may help.

On the Syslog page all incoming Syslog messages are shown except for those of the firewall. They have an own page Firewall. In order for this to work the varable `OPT_SYSLOGD` in config/base.txt has to be set to 'yes'. The variable `SYSLOGD_DEST` must contain the clients IP (i.e. `SYSLOGD_DEST`='@100.100.100.100 – of course with the real IP of the clients). Syslog message and according date, time, IP of the Senders and priority will be shown.

Firewall messages are displayed on an own page Firewall to be better readable. `OPT_KLOGD` must be set to 'yes' in config/base.txt in addition.

7.2.11. News Page

If the option is activated in imonc's config news from the fli4l homepage are shown here directly in imonc. Via http protocol the URL http://www.fli4l.de/german/news.xml will be loaded. The five newest opt-packages are shown here as well. For this the URL http://www.fli4l.de/german/imonc_opt_show.php will be queried. In the status bar the headers of the news will be shown alternatingly.

7.3. Unix/Linux-Client imonc

There are 2 different versions for Linux: a text-based imonc) and a graphical user interface version(ximonc). The source of ximonc can be found under the directory src. The documentation for ximonc will only be available in the 1.5-Final-Version. An experienced Linux-User should have no problems with the source.

Let's limit to the text-based version. This is a curses based program, thus it has no graphical interface. The source lies under the directory unix.

Installation:

```
cd unix
make install
```

imonc will be installed to /usr/local/bin.

Command line parameters:

```
imonc hostname
```

hostname can be the name or the IP address of the fli4l router, e.g.

```
imonc fli4l
```

imonc shows the following:

- Date/Time of the fli4l router
- Momentarily configured route

- Default-Route-Circuits
- ISDN channels

  Status : Calling/Online/Offline

  Name : Phone number of the peer or the circuit-name

  Time : Online time

  Charge-Time : Online time considering the charge interval

  Charge : The actual charge

Possible commands:

Nr	Command	Meaning
0	quit	Quit program
1	enable	Activate
2	disable	Deactivate
3	dial	Dial
4	hangup	Hang up
5	reboot	Reboot
6	timetable	Output timetable
7	dflt route	Set Default-Route-Circuit
8	add channel	Add 2. channel
9	rem channel	Remove 2. channel

Detailed information on every command:

0 – quit The connection to the imond server is terminated and the program quits.

1 – enable All circuits are set to dialmode "auto". This is the default state after boot. It results in fli4l dialing automatically on demand as soon as it receives a request by a host from the LAN.

2 – disable All circuits are set to dialmode "off". This means fli4l is virtually "dead" until it is revived by the enable command.

3 – dial Manual dial using the Default-Route-Circuit. You won't need this normally as fli4l normally dials automatically.

4 – hangup Manual hangup. You can make fli4l hangup before it does it automatically.

5 – reboot fli4l is rebooted. Pretty unnecessary command ...

6 – timetable The timetable for the Default-Route-Circuits is printed out. Example: see above.

7 – default route circuit Manually changing the default route circuit can make sense, if you want to disable the automatic LC routing of fli4l for a while, as some providers will only let you access your email if you are dialed in to their servers.

8 – add channel The second ISDN channel is manually added. Prerequisite: `ISDN_CIRC_x_BUNDLING` is set to 'yes'.

9 – remove channel Removes the second ISDN channel. See also “add channel”.

Apart from that, the same annotations as for the windows client `imonc.exe` apply.

A little remark: From fli4l-1.4 on, it is possible, to install a “minimalistic” imon client on the fli4l router itself using `OPT_IMONC`=’yes’ in package `TOOLS`.

You will be able to change some settings, e.g. routing etc. on the fli4l console locally. But Beware: This mini-imonc will only work on the fli4l router itself! On a Linux or Unix client you should always use the “big brother” unix/imonc.

8. Documentation for Developers

8.1. Common Rules

In order to include a new package in the OPT database on the fli4l homepage some rules must be obeyed. Packages that do not comply with these rules may be removed from the database without warning.

1. *No* file copy actions by the user! fli4l provides a sophisticated system to include the fli4l packages into the installation archives. All files that should go to the router are located in `opt/`.

2. Pack and compress packets properly: packages must be constructed in a way to allow effortless unpacking into the fli4l directory.

3. Packages must be *completely* configurable via the config file. Further editing of configuration files shall not be required by the user. Keep difficult decisions away from the users and move them to another part at the end of the configuration file. *Note: ONLY MODIFY IF YOU KNOW WHAT YOU DO.*

4. Another hint for config files: by its name it has to be seen clearly which OPT it belongs to. For example `OPT_HTTPD` contains the variables `OPT_HTTPD`, `HTTPD_USER_N`, a.s.o.

5. Please built binaries as small as possible! If you compile them yourself in FBR please remember to deactive any unneeded feature.

6. Check for Copyrights! If using template files please keep in mind to change the Copyright accordingly. This applies also for config-, check- and Opt text files. Replace the Copyright with your own name. A documentation that was only copied of course has to keep the Copyright of the orginal author!

7. Please use only free formats as archive types. These are:

   - ZIP (`.zip`)
   - GZIP (`.tgz` or `.tar.gz`)

   Please don't use other formats like RAR, ACE, Blackhole, LHA ... Windows-Installer files (`.msi`) or self-extracting archives and installers (`.exe`) should *not* be used.

8.2. Compiling Programs

The environment needed for compiling programs is available in the separate package "src". There it is also documented how to compile own programs for fli4l.

8.3. Module Concept

As of version 2.0 fli4l is split into modules (packages), i.e.

- fli4l-3.10.5 <— The Base Package
- dns-dhcp
- dsl
- isdn
- sshd
- and much more...

With the base package fli4l acts as a pure Ethernet router. For ISDN and/or DSL the packages isdn and/or dsl have to be unpacked to the fli4l directory. The same applies for the other packages.

8.3.1. mkfli4l

Depending on the current configuration a file called `rc.cfg` and two archives `rootfs.img` and `opt.img` will be generated which contain all required configuration informations and files. These files are generated using `mkfli4l` which reads the individual package files and checks for configuration errors.

`mkfli4l` will accept the parameters listed in table 8.1. If omitted the default values noted in brackets are used. A complete list of all options (Table 8.1) is displayed when executing

```
mkfli4l -h
```

.

8.3.2. Structure

A package can contain multiple OPTs, if it contains only one, however, it is appropriate to name the package like the OPT. Below `<PACKAGE>` is to be replaced by the respective package name. A package consists of the following parts:

- Administrative Files
- Documentation
- Developer Documentation
- Client Programs
- Source Code
- More Files

The individual parts are described in more detail below.

Table 8.1.: Parameters for `mkfli4l`

Option	Meaning
-c, --config	Declaration of the directory `mkfli4l` will scan for package config files (default: config)
-x, --check	Declaration of the directory `mkfli4l` will scan for files needed for package error checking (`<package>.txt`, `<package>.exp` and `<package>.ext`; default: check)
-l, --log	Declaration of the log file to which `mkfli4l` will log error messages and warnings (default: `img/mkfli4l.log`)
-p, --package	Declaration of the packages to be checked, this option may be used more than once in case of a desired check for several packages in conjunction. If using -p, however, the file `<check_dir>/base.exp` will always be read first to provide the common regular expressions provided by the base package. Hence, this file must exist.
-i, --info	Provides information on the check (which files are read, which tests are run, which uncommon things happened during the review process)
-v, --verbose	More verbose variant of option -i
-h, --help	Displays the help text
-d, --debug	Allows for debugging the review process. This is meant to be a help for package developers wishing to know in detail how the process of package checking is working.

Debug Option	Meaning
check	show check process
zip-list	show generation of zip list
zip-list-skipped	show skipped files
zip-list-regexp	show regular expressions for ziplist
opt-files	check all files in `opt/<package>.txt`
ext-trace	show trace of extended checks

8.3.3. Configuration of Packages

The user's changes to the package's configuration are made in the file `config/<PACKAGE>.txt`. All the OPT's variables should begin with the name of the OPT, for example:

```
#------------------------------------------------------------------
# Optional package: TELNETD
#------------------------------------------------------------------
OPT_TELNETD='no'        # install telnetd: yes or no
TELNETD_PORT='23'       # telnet port, see also FIREWALL_DENY_PORT_x
```

An OPT should be prefixed by a header in the configuration file (see above). This increases readability, especially as a package indeed can contain multiple OPTs. Variables associated to the OPT should — again in the interest of readability — not be indented further. Comments and blank lines are allowed, with comments always starting in column 33. If a variable including its content has more than 32 characters, the comment should be inserted with a row offset, starting in column 33. Longer comments are spread over multiple lines, each starting at column 33. All this increases easy review of the configuration file.

All values following the equal sign must be enclosed in quotes[1] not doing so can lead to problems during system boot.

Activated variables (see below), will be transferred to `rc.cfg`, everything else will be ignored. The only exceptions are variables by the name of `<PACKAGE>_DO_DEBUG`. These are used for debugging and are transferred as is.

8.3.4. List of Files to Copy

The file `opt/<PACKAGE>.txt` contains instructions that describe

- which files are owned by the OPT,
- the preconditions for inclusion in the `opt.img` resp. `rootfs.img`,
- what User ID (uid), Group ID (gid) and rights will be applied to files,
- which conversions have to be made before inclusion in the archive.

Based on this information `mkfli4l` will generate the archives needed.

Blank lines and lines beginning with "#" are ignored. In one of the first lines the version of the package file format should be noted as follows:

```
<first column>      <second column> <third column>
opt_format_version  1                  -
```

The remaining lines have the following syntax:

```
<first column>  <second column> <third column> <columns following>
Variable        Value           File           Options
```

[1] Both single and double quotes are valid. You can hence write `FOO='bar'` as well as `FOO="bar"`. The use of double quotes should be an exception and you should previously inform about how an *nix shell uses single and double quotes.

1. The first column contains the name of a variable which triggers inclusion of the file referenced in the third column depending on its value in the package's config file. The name of a variable may appear in the first column as often as needed if multiple files depend on it. Any variable that appears in the file `opt/<PACKAGE>.txt` is marked by `mkfli4l`.

   If multiple variables should be tested for the same value a list of variables (separated by commas) can be used instead. It is sufficient in this case if at least *one* variable contains the value required in the second column. It is important *not* to use spaces between the individual variables!

   In OPT variables (ie variables that begin with `OPT_` and typically have the type `YESNO`), the prefix "`OPT_`" can be omitted. It does not matter whether variables are noted in upper- or lowercase (or mixed).

2. The second column contains a value. If the variable in the first column is identical with this value and is activated too (see below), the file referenced in the third column will be included. If the first column contains a %-variable it will be iterated over all indices and checked whether the respective variable matches the value. If this is the case copying will be executed. In addition, the copy process based on the current value of the variable will be logged.

   It is possible to write a "!" in front of the value. In this case, the test is negated, meaning the file is only copied if the variable does *not* contain the value.

3. In the third column a file name is referenced. The path must be given relative to the `opt` directory. The file must exist and be readable, otherwise an error is raised while generating the boot medium and the build process is aborted.

   If the file name is prefixed with a "`rootfs:`" the file is included in the list of files to be copied to the RootFS. The prefix will be stripped before.

   If the file is located below the current configuration directory it is added to the list of files to be copied from there, otherwise the file found below opt is taken. Those files are not allowed to have a `rootfs:` prefix.

   If the file to copy is a kernel module the actual kernel version may be substituted by `${KERNEL_VERSION}`. `mkfli4l` will then pick the version from the configuration and place it there. Using this you may provide modules for several kernel versions for the package and the module matching the current kernel version will be copied to the router. For kernel modules the path may be omitted, `mkfli4l` will find the module using `modules.dep` and `modules.alias`, see the section "Automatically Resolving Dependencies for Kernel Modules" (Page 289).

4. the other columns may contain the options for owner, group, rights for files and conversion listed in table 8.2.

Some examples:

- copy file if `OPT_TELNETD='yes'`, set its uid/gid to root and the rights to 755 (`rwxr-xr-x`)

```
telnetd     yes    files/usr/sbin/in.telnetd mode=755
```

Table 8.2.: Options for Files

Option	Meaning	Default Value
type=	Type of the Entry: *local* Filesystem Object *file* File *dir* Directory *node* Device *symlink* (symbolic) Link This option has to be placed in front when given. The type "local" represents the type of an object existing in the file system and hence matches "file", "dir", "node" or "symlink" (depends). All other types except for "file" can create entries in the archive that do not have to exist in the local file system. This can i.e. be used to create devices files in the RootFS archive.	local
uid=	The file owner, either numeric or as a name from passwd	root
gid=	File group, either numeric or as a name from group	root
mode=	Access rights	Files and Devices: `rw-r--r--` (644) Directories: `rwxr-xr-x` (755) Links: `rwxrwxrwx` (777)
flags= (type=file)	Conversions before inclusion in the archive: *utxt* Conversion to *nix format *dtxt* Conversion to DOS format *sh* Shell-Skript: Conversion to *nix format, stripping of superfluous chars	
name=	Alternative name for inclusion of the entry in the archive	
devtype= (type=node)	Descibes the type of the device ("c" for character and "b" für block oriented devices). Has to be placed in second position.	
major= (type=node)	Decribes the so-called "Major" number of the device file. Has to be placed in third position.	
minor= (type=node)	Decribes the so-called "Minor" number of the device file. Has to be placed in fourth position.	
linktarget= (type=symlink)	Describes the target of the symbolic link. Has to be placed in second position.	

- copy file, set uid/gid to root, the rights to 555 (`r-xr-xr-x`) and convert the file to *nix format while stripping all superfluous chars

```
base    yes     etc/rc0.d/rc500.killall mode=555 flags=sh
```

- copy file if `PCMCIA_PCIC='i82365'`, set uid/gid to root and the rights to 644 (`rw-r--r--`)

```
pcmcia_pcic i82365 files/lib/modules/${KERNEL_VERSION}/pcmcia/i82365.ko
```

- copy file if one of the `NET_DRV_%` variables matches the second field, set uid/gid to root and the rights to 644 (`rw-r--r--`)

```
net_drv_%   3c503  3c503.ko
```

- copy file if the variable `POWERMANAGEMENT` does *not* contain the value "none":

```
powermanagement !none etc/rc.d/rc100.pm mode=555 flags=sh
```

- copy file if any of the OPT variables `OPT_MYOPTA` or `OPT_MYOPTB` contains the value "yes":

```
myopta,myoptb yes files/usr/local/bin/myopt-common.sh mode=555 flags=sh
```

  This example is only an abbreviation for:

```
myopta yes files/usr/local/bin/myopt-common.sh mode=555 flags=sh
myoptb yes files/usr/local/bin/myopt-common.sh mode=555 flags=sh
```

  And the latter is a shorthand notation for:

```
opt_myopta yes files/usr/local/bin/myopt-common.sh mode=555 flags=sh
opt_myoptb yes files/usr/local/bin/myopt-common.sh mode=555 flags=sh
```

- copy file `opt/files/usr/bin/beep.sh` to the RootFS archive, but rename it to `bin/beep` before:

```
base yes rootfs:files/usr/bin/beep.sh mode=555 flags=sh name=bin/beep
```

The files will be copied only if the above conditions are met and `OPT_PACKAGE='yes'` of the corresponding package is set. What OPT variable is referenced is decribed in the file `check/<PACKAGE>.txt`.

If a variable is referenced in a package that is not defined by the package itself, it may happen that the corresponding package is not installed. This would result in an error message from `mkfli4l`, as it expects that all of the variables referenced by `opt/<PACKAGE>.txt` are defined.

To handle this situation correctly the "weak" declaration has been introduced. It has the following format:

```
weak        <Variable>    -
```

By this the variable it is defined (if not already existing) and its value is set to "undefined". Please note that the prefix "`OPT_`" *must* be provided (if existing) because else a variable *without* this prefix will be created.

An example taken from `opt/rrdtool.txt`:

```
weak opt_openvpn -
[...]
openvpn    yes    files/usr/lib/collectd/openvpn.so
```

Without the `weak` definition `mkfli4l` would display an error message when using the package "rrdtool" while the "openvpn" package is not activated. By using the `weak` definition no error message is raised even in the case that the "openvpn" package does not exist.

Files adapted by Configuration

In some situations it is desired to replace original files with configuration-specific files for inclusion in the archive, i.e. host keys, own firewall scripts, ... `mkfli4l` supports this scenario by checking whether a file can be found in the configuration directory and, if so, including this one instead in the file list for `opt.img` resp. `rootfs.img`.

Another option to add configuration-specific files to an archive is decribed in the section Extended Checks of the Configuration (Page 306).

Automatically Resolving Dependencies for Kernel Modules

Kernel modules may depend on other kernel modules. Those must be loaded before and therefore also have to be added to the archive. `mkfli4l` resolves this dependencies based on `modules.dep` and `modules.alias` (two files generated during the kernel build), automatically including all required modules in the archives. Thus, for example the following entry

```
net_drv_%   ne2k-pci    ne2k-pci.ko
```

triggers that both 8390.ko and crc32.ko are included in the archive because ne2k_pci depends on both of them.

The necessary entries from `modules.dep` and `modules.alias` are included in the RootFS and can be used by `modprobe` for loading the drivers.

8.3.5. Checking Configuration Variables

By the help of `check/<PACKAGE>.txt` the content of variables can be checked for validity. In earlier version of the program `mkfli4l` this check was hard coded there but it was outsourced to the check files in the course of modularizing fli4l. This file contains a line for each variable in the config files. These lines consist of four to five columns which have the following functions:

1. Variable: this column specifies the name of the configuration file variable to check. If this is an *array variable*, it can appear multiple times with different indices, so instead of the index number a percent sign (%) is added to the variable name. It is always used as "`_%_`" in the middle of a name resp. "`_%`" at the end of a name. The name may contain more than one percent sign allowing the use of multidimensional arrays. It is recommended (but not mandatory) to add some text between the percent signs to avoid weird names like "`FOO_%__%`".

   Often the problem occurs that certain variables describe options that are needed only in some situations. Therefore variables may be marked as optional. Optional variables are identified by the prefix "+". They may then exist, but do not have to. Arrays can also use a "++" prefix. Prefixed with a "+" the array can exist or be entirely absent. Prefixed with a"++" in addition some elements of the array may be missing.

2. `OPT_VARIABLE`: This column assigns the variable to a specific OPT. The variable is checked for validity only if the OPT variable is set to "yes". If there is no OPT variable a "-" indicates this. In this case, the variable must be defined in the configuration file, unless a default value is defined (see below). The name of the OPT variable may be arbitrary but should start with the prefix "`OPT_`".

If a variable does not depend on any OPT variables, it is considered *active*. If it is depending on an OPT variable, it is precisely active if

- its OPT variable is active and
- its OPT variable contains the value "yes".

In all other cases the variable is inactive.

Hint: Inactive OPT variables will be set back to "no" by `mkfli4l` if set to "yes" in the configuration file, an appropriate warning will be generated then (i.e. `OPT_Y='yes'` is ignored, because `OPT_X='no'`). For transitive dependency chains (`OPT_Z` depends on `OPT_Y` which in turn depends on `OPT_X`) this will only work reliable, if the names of all OPT-variables start with "`OPT_`".

3. `VARIABLE_N`: If the first column contains a variable with a "%" in its name, it indicates the number of occurrences of the variable (the so-called *N-variable*). In case of a multi-dimensional variable, the occurences of the last index are specified. If the variable depends on a certain OPT, the N-variable must be dependant on the same or no OPT. If the variable does not depend on any OPT, the N-variable also shouldn't. If no N-variable exists, specify "-" to indicate that.

   For compatibility with future versions of fli4l the variable specified here *must* be identical with the variable in `OPT_VARIABLE` where the last "%" is replaced by an "N" and everything following is removed. An array `HOST_%_IP4` must have the N-Variable `HOST_N` assigned and an array `PF_USR_CHAIN_%_RULE_%` hence the N-variable `PF_USR_CHAIN_%_RULE_N`, and this N-variable itself is an array variable with the corresponding N-variable `PF_USR_CHAIN_N`. *All other namings of the N variables will be incompatible with future versions of fli4l!*

4. `VALUE`: This column provides the values a variable can hold. For example the following settings are possible:

Name	Meaning
`NONE`	No error checking will be done
`YESNO`	The variable must be "yes" or "no"
`NOTEMPTY`	The variable can't be empty
`NOBLANK`	The variable can't contain spaces
`NUMERIC`	The variable must be numeric
`IPADDR`	The variable must be an IP address
`DIALMODE`	The variable must be "on", "off" or "auto"

   I values are prefixed by "`WARN_`" an illegal content will not raise an error message and abort the build by `mkfli4l`, but only display a warning.

   The possible checks are defined by regular expressions in `check/base.exp`. This file may be extended and now contains some new checking routines, for example: `HEX`, `NUMHEX`, `IP_ROUTE`, `DISK` and `PARTITION`.

   The number of expressions may be extended at any time for the future needs of package developers. Provide feedback!

   In addition, regular expressions can also be directly defined in the check-files, even relations to existing expressions can be made. Instead of `YESNO` you could, for example also write

```
RE:yes|no.
```

This is useful if a test is performed only once and is relatively easy. For more details see the next chapter.

5. Default Setting: In this column, an optional default value for the variables can be defined in the case that the variable is not specified in the configuration file.

   Hint: At the moment this does not work for array variables. Additionally, the variable can't be optional (no "+" in front of the variable name).

   Example:

   ```
   OPT_TELNETD     -       -       YESNO     "no"
   ```

   If `OPT_TELNETD` is missing in the config file, "no" will be assumed and written as a value to `rc.cfg`.

The percent sign thingie is best decribed with an example. Let's assume `check/base.txt` amongst others has the following content:

```
NET_DRV_N           -                   -                   NUMERIC
NET_DRV_%           -                   NET_DRV_N           NONE
NET_DRV_%_OPTION    -                   NET_DRV_N           NONE
```

This means that depending on the value of `NET_DRV_N` the variables `NET_DRV_N`, `NET_DRV_1_OPTION`, `NET_DRV_2_OPTION`, `NET_DRV_3_OPTION`, a.s.o. will be checked.

8.3.6. Own Definitions for Checking the Configuration Variables

Introduction of Regular Expressions

In version 2.0 only the above mentioned value ranges for variable checks existed: `NONE`, `NOTEMPTY`, `NUMERIC`, `IPADDR`, `YESNO`, `NOBLANK`, `DIALMODE`. Checking was hard-coded to `mkfli4l`, not expandable and restricted to essential "data types" which could be evaluated with reasonable efforts.

As of version 2.1 this checking has been reimplemented. The aim of the new implementation is a more flexible testing of variables, that is also able to examine more complex expressions. Therefore, regular expressions are used that can be stored in one or more separate files. This on one hand makes it possible to examine variables that are not checked for the moment and on the other hand, developers of optional packages can now define own terms in order to check the configuration of their packages.

A description of regular expressions can be found via "man 7 regex" or i.e. here: http://unixhelp.ed.ac.uk/CGI/man-cgi?regex+7.

Specification of Regular Expressions

Specification of regular expressions can be accomplished in two ways:

1. Package specific exp files `check/<PACKAGE>.exp`

   This file can be found in the `check` directory and has the same name as the package containing it, i.e. `check/base.exp`. It contains definitions for expressions that can

be referenced in the file `check/<PACKAGE>.txt`. `check/base.exp` for example at the moment contains definitions for the known tests and `check/isdn.exp` a definition for the variable `ISDN_CIRC_x_ROUTE` (the absence of this check was the trigger for the changes).

The syntax is as follows (again, double quotes can be used if needed):

```
<Name> = '<Regular Expression>' : '<Error Message>'
```

as an example `check/base.exp`:

```
NOTEMPTY = '.*[^ ]+.*'          : 'should not be empty'
YESNO    = 'yes|no'             : 'only yes or no are allowed'
NUMERIC  = '0|[1-9][0-9]*'      : 'should be numeric (decimal)'
OCTET    = '1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5]'
         : 'should be a value between 0 and 255'
IPADDR   = '((RE:OCTET)\.){3}(RE:OCTET)' : 'invalid ipv4 address'
EIPADDR  = '()|(RE:IPADDR)'
         : 'should be empty or contain a valid ipv4 address'
NOBLANK  = '[^ ]+'              : 'should not contain spaces'
DIALMODE = 'auto|manual|off'    : 'only auto, manual or off are allowed'
NETWORKS = '(RE:NETWORK)([[:space:]]+(RE:NETWORK))*'
         : 'no valid network specification, should be one or more
            network address(es) followed by a CIDR netmask,
            for instance 192.168.6.0/24'
```

The regular expressions can also include already existing definitions by a reference. These are then pasted to substitute the reference. This makes it easier to construct regular expressions. The references are inserted by '(RE: Reference)'. (See the definition of the term `NETWORKS` above for an appropriate example.)

The error messages tend to be too long. Therefore, they may be displayed on multiple lines. The lines afterwards always have to start with a space or tab then. When reading the file `check/<PACKAGE>.exp` superfluous whitespaces are reduced to one and tabs are replaced by spaces. An entry in `check/<PACKAGE>.exp` could look like this:

```
NUM_HEX         = '0x[[:xdigit:]]+'
                : 'should be a hexadecimal number
                   (a number starting with "0x")'
```

2. Regular expressions directly in the check file `check/<PACKAGE>.txt`

   Some expressions occur but once and are not worth defining a regular expression in a `check/<PACKAGE>.exp` file. You can simply write this expression to the check file for example:

```
# Variable      OPT_VARIABLE    VARIABLE_N      VALUE
MOUNT_BOOT      -               -               RE:ro|rw|no
```

   `MOUNT_BOOT` can only take the value “ro”, “rw” or “no”, everything else will be denied.

   If you want to refer to existing regular expressions, simply add a reference via ‘(RE:...)”. Example:

```
# Variable      OPT_VARIABLE    VARIABLE_N      VALUE
LOGIP_LOGDIR    OPT_LOGIP       -               RE:(RE:ABS_PATH)|auto
```

Expansion of Existing Regular Expressions

If an optional package adds an additional value for a variable which will be examined by a regular expression, then the regular expression has to be expanded. This is done simply by defining the new possible values by a regular expression (as described above) and complement the existing regular expression in a separate `check/<PACKAGE>.exp` file. That an existing expression is modified is indicated by a leading "+". The new expression complements the existing expression by appending the new value to the existing value(s) as an alternative. If another expression makes use of the complemented expression, the supplement is also there. The specified error message is simply appended to the end of the existing one.

Using the Ethernet driver as an example this could look like here:

- The base packages provides a lot of Ethernet drivers and checks the variable `NET_DRV_x` using the regular expression `NET_DRV`, which is defined as follows:

```
NET_DRV         = '3c503|3c505|3c507|...'
                : 'invalid ethernet driver, please choose one'
                  ' of the drivers in config/base.txt'
```

- The package "pcmcia" provides additional device drivers, and hence has to complement `NET_DRV`. This is done as follows:

```
PCMCIA_NET_DRV = 'pcnet_cs|xirc2ps_cs|3c574_cs|...' : ''
+NET_DRV       = '(RE:PCMCIA_NET_DRV)' : ''
```

Now PCMCIA drivers can be chosen in addition.

Extend Regular Expressions in Relation to `YESNO` Variables

If you have extended `NET_DRV` with the PCMCIA drivers as shown above, but the package "pcmcia" has been deactivated, you still could select a PCMCIA driver in `config/base.txt` without an error message generated when creating the archives. To prevent this, you may let the regular expression depend on a `YESNO` variable in the configuration. For this purpose, the name of the variable that determines whether the expression is extended is added with brackets immediately after the name of the expression. If the variable is active and has the value "yes", the term is extended, otherwise not.

```
PCMCIA_NET_DRV       = 'pcnet_cs|xirc2ps_cs|3c574_cs|...' : ''
+NET_DRV(OPT_PCMCIA) = '(RE:PCMCIA_NET_DRV)' : ''
```

If specifying `OPT_PCMCIA='no'` and using i.e. the PCMCIA driver `xirc2ps_cs` in `config/base.txt`, an error message will be generated during archive build.

Hint: This does *not* work if the variable is not set explicitely in the configuration file but gets its value by a default setting in `check/<PACKAGE>.txt`. In this case the variable hence has to be set explicitely and the default setting has to be avoided if necessary.

Extending Regular Expressions Depending on other Variables

Alternatively, you may also use arbitrary values of variables as conditions, the syntax looks like this:

```
+NET_DRV(KERNEL_VERSION=~'^3\.14\..*$') = ...
```

If `KERNEL_VERSION` matches the given regular expression (if any of the kernels of the 3.14 line is used) then the list of network driver allowed is extended with the drivers mentioned.

Hint: This does *not* work if the variable is not set explicitely in the configuration file but gets its value by a default setting in `check/<PACKAGE>.txt`. In this case the variable hence has to be set explicitely and the default setting has to be avoided if necessary.

Error Messages

If the checking process detects an error, an error message of the following kind is displayed:

```
Error: wrong value of variable HOSTNAME: '' (may not be empty)
Error: wrong value of variable MOUNT_OPT: 'rx' (user supplied regular expression)
```

For the first error, the term was defined in a `check/<PACKAGE>.exp` file and an explanation of the error is displayed. In the second case the term was specified directly in a `check/<PACKAGE>.txt` file, so there is no additional explanation of the error cause.

Definition of Regular Expressions

Regular expressions are defined as follows:

Regular expression: One or more alternatives, separated by '|', i.e. "ro|rw|no". If one option matches, the whole term matches (in this case "ro", "rw" and "no" are valid expressions).

An alternative is a concatenation of several sections that are simply added.

A section is an "atom", followed by a single "*", "+", "?" or "{min, max}". The meaning is as follows:

- "a*" — as many "a"s as wished (allows also no "a" is existing at all)
- "a+" — at least one"a"
- "a?" — none or one "a"
- "a{2,5}" — two to five "a"s
- "a{5}" — exactly five "a"s
- "a{2,}" — at least two "a"s
- "a{,5}" — a maximum of five "a"s

An "atom" is a

- regular expression enclosed in brackets, for example "(a|b)+" matches any string containing at least one "a" or "b", up to an arbitrary number and in any order

- an empty pair of brackets stands for an "empty" expression
- an expression in square brackets "[]" (see below)
- a dot ".", matching an arbitrary character, for example a ".+" matches any string containing at least one char
- a "^" represents the beginning of a line, for example a "^a.*" matches a string beginning with an "a" followed by any char like in "a" or "adkadhashdkash"
- a "$" represents the end of a line
- a "\" followed by one of the special characters `^ . [ $ ( ) | * + ? { \` stands for the second char without its special meaning
- a normal char matches exactly this char, for example "a" matches exactly an "a".

An expression in square brackets indicates the following:

- "x-y" — matches any char inbetween "x" and "y", for example "[0-9]" matches all chars between "0" and "9"; "[a-zA-Z]" symbolizes all chars, either upper- or lowercase.
- "^x-y" — matches any char *not* contained in the given interval, for example "[^0-9]" matches all chars *except* for digits.
- "[:*character-class*:]" — matches a char from *character-class*. Relevant standard character classes are: `alnum`, `alpha`, `blank`, `digit`, `lower`, `print`, `punct`, `space`, `upper` and `xdigit`. I.e. "[[:alpha:]]" stands for all upper- or lowercase chars and hence is identical with "[[:lower:][:upper:]]".

Examples for regular Expressions

Let's have a look at some examples!

`NUMERIC`: A numeric value consists of at least one, but otherwise any number of digits. "At least one" is expressed with a "+", one digit was already in an example above. So this results in:

```
NUMERIC = '[0-9]+'
```

or alternatively

```
NUMERIC = '[[:digit:]]+'
```

`NOBLANK`: A value that does not contain spaces, is any char (except for the char "space") and any number of them:

```
NOBLANK = '[^ ]*'
```

or, if the value is not allowed to be empty:

```
NOBLANK = '[^ ]+'
```

`IPADDR`: Let's have a look at an example with an IP4-address. An ipv4 address consists of four "Octets", divided by dots ("."). An octet is a number between 0 and 255. Let's define an octet at first. It may be

a number between 0 and 9:	[0-9]
a number between 10 and 99:	[1-9][0-9]
a number between 100 and 199:	1[0-9][0-9]
a number between 200 and 249:	2[0-4][0-9]
a number between 250 and 255:	25[0-5]

All are alternatives hence we concatenate them with "|" forming one expression: "[0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5]" and get an octet. Now we compose an IP4 address, four octets divided by dots (the dot must be masked with a *backslash*, because else it would represent an arbitrary char). Based on the syntax of an exp-file it would look like this:

```
OCTET  = '[0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5]'
IPADDR = '((RE:OCTET)\.){3}(RE:OCTET)'
```

Assistance for the Design of Regular Expressions

If you want to design and test regular expressions, you can use the "regexp" program located in the `unix` or `windows` directory of the package "base". It accepts the following syntax:

```
usage: regexp [-c <check dir>] <regexp> <string>
```

The parameters explained in short:

- `<check dir>` is the directory containing check and exp files. These are read by "regexp" to use expressions already defined there.
- `<regexp>` is the regular expression (enclosed in `'...'` or `"..."` if in doubt, with double quotes needed only if single quotes are used in the expression itself)
- `<string>` is the string to be checked

This may for example look like here:

```
./i586-linux-regexp -c ../check '[0-9]' 0
adding user defined regular expression='[0-9]' ('^([0-9])$')
checking '0' against regexp '[0-9]' ('^([0-9])$')
'[0-9]' matches '0'

./i586-linux-regexp -c ../check '[0-9]' a
adding user defined regular expression='[0-9]' ('^([0-9])$')
checking 'a' against regexp '[0-9]' ('^([0-9])$')
regex error 1 (No match) for value 'a' and regexp '[0-9]' ('^([0-9])$')

./i586-linux-regexp -c ../check IPADDR 192.168.0.1
using predefined regular expression from base.exp
adding IPADDR='((RE:OCTET)\.){3}(RE:OCTET)'
 ('^(((1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])\.){3}(1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5]))$')
'IPADDR' matches '192.168.0.1'
```

```
./i586-linux-regexp -c ../check IPADDR 192.168.0.256
using predefined regular expression from base.exp
adding IPADDR='((RE:OCTET)\.){3}(RE:OCTET)'
 ('^(((1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])\.){3}(1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5]))$')
regex error 1 (No match) for value '192.168.0.256' and regexp
 '((RE:OCTET)\.){3}(RE:OCTET)'
(unknown:-1) wrong value of variable cmd_var: '192.168.0.256' (invalid ipv4 address)
```

8.3.7. Extended Checks of the Configuration

Sometimes it is necessary to perform more complex checks. Examples of such complex things would be i.e. dependencies between packages or conditions that must be satisfied only when variables take certain values. For example if a PCMCIA ISDN adapter is used the package "pcmcia" has to be installed, too.

In order to perform these checks you may write small tests to `check/<PACKAGE>.ext` (also called ext-script). The language consists of the following elements:

1. Keywords:
   - Control Flow:
     - `if (`*`expr`*`) then` *`statement`* `else` *`statement`* `fi`
     - `foreach` *`var`* `in` *`set_var`* `do` *`statement`* `done`
     - `foreach` *`var`* `in` *`set_var_1 ... set_var_n`* `do` *`statement`* `done`
     - `foreach` *`var`* `in` *`var_n`* `do` *`statement`* `done`
   - Dependencies:
     - `provides` *`package`* `version` *`x.y.z`*
     - `depends on` *`package`* `version` *`x1.y1 x2.y2.z2 x3.y3 ...`*
   - Actions:
     - `warning "`*`warning`*`"`
     - `error "`*`error`*`"`
     - `fatal_error "`*`fatal error`*`"`
     - `set` *`var`* `=` *`value`*
     - `crypt (`*`variable`*`)`
     - `stat (`*`filename, res`*`)`
     - `fgrep (`*`filename, regex`*`)`
     - `split (`*`string, set_variable, character`*`)`
2. Data Types: strings, positive integers, version numbers
3. Logical Operations: `<`, `==`, `>`, `!=`, `!`, `&&`, `||`, `=~`, `copy_pending`, `samenet`, `subnet`

Data Types

Concerning data types please note that variables, based on the associated regular expression are permanently assigned to a data type:

- Variables, starting with type "NUM" are numeric and contain positive integers
- Variables representing an N-variable for any kind of array are numeric as well
- all other variables are treated as strings

This means, among other things, that a variable of type ENUMERIC can *not* be used as an index when accessing an array variable, even if you have checked at first that it is not empty. The following code thus does not work as expected:

```
# TEST should be a variable of type ENUMERIC
if (test != "")
then
    # Error: You can't use a non-numeric ID in a numeric
    #         context. Check type of operand.
    set i=my_array[test]
    # Error: You can't use a non-numeric ID in a numeric
    #         context. Check type of operand.
    set j=test+2
fi
```

A solution for this problem is offered by split (Page 305):

```
if (test != "")
then
    # all elemente of test_% are numeric
    split(test, test_%, ' ', numeric)
    # OK
    set i=my_array[test_%[1]]
    # OK
    set j=test_%[1]+2
fi
```

Substitution of Strings and Variables

At various points strings are needed, such as when a Warning (Page 301) should be issued. In some cases described in this documentation, such a string is scanned for variables. If found, these are *replaced* by their contents or other attributes. This replacement is called *variable substitution*.

This will be illustrated by an example. Assume this configuration:

```
# config/base.txt
HOSTNAME='fli4l'
# config/dns_dhcp.txt
HOST_N='1' # Number of hosts
HOST_1_NAME='client'
HOST_1_IP4='192.168.1.1'
```

Then the character strings are rewritten as follows, if variable substitution is active in this context:

```
"My router is called $HOSTNAME"
# --> "My router is called fli4l"
"HOSTNAME is part of the package %{HOSTNAME}"
# --> "HOSTNAME is part of the package base"
"@HOST_N is $HOST_N"
# --> " # Number of hosts is 1"
```

As you can see, there are basically three options for replacement:

- `$<Name>` resp. `${<Name>}`: Replaces the variable name with the contents of the variable. This is the most common form of replacement. The name must be enclosed in `{...}` if in the string it is directly followed by a char that may be a valid part of a variable name (a letter, a digit, or an underscore). In all other cases, the use of curly brackets is possible, but not mandatory.
- `%<Name>` resp. `%{<Name>}`: Replaces the variable name with the name of the package in which the variable is defined. This does *not* work with variables assigned in the script via `set` (Page 301) or with counting variables of a `foreach`-loop (Page 307) since such variables do not have a package and their syntax is different.
- `@<Name>` resp. `@{<Name>}`: Replaces the variable name with the comment noted in the configuration after the variable. Again, this does not make sense for variables defined by the script.

Hint: Elements of array variables can *not* be integrated into strings this way, because there is no possibility to provide an index.

In general, only *constants* can be used for variable substitution, strings that come from a variable remain unchanged. An example will make this clear - assume the following configuration:

```
HOSTNAME='fli4l'
TEST='${HOSTNAME}'
```

This code:

```
warning "${TEST}"
```

leads to the following output:

```
Warning: ${HOSTNAME}
```

It will *not* display:

```
Warning: fli4l
```

In the following sections it will be explicitly noted under which conditions strings are subject of variable substitution.

Definition of a Service with an associated Version Number: `provides`

For instance, an OPT may declare that it provides a Printer service or a Webserver service. Only one package can provide a certain service. This prevents i.e. that two web servers are installed in parallel, which is not possible for obvious reasons, since the two servers would both register port 80. In addition, the current version of the service is provided so that updates can be triggered. The version number consists of two or three numbers separated by dots, such as "4.0" or "2.1.23".

Services typically originate from OPTs, not from packages. For example the package "tools" has a number of programs that each have their own `provides` statement defined if activated by `OPT_...='yes'`.

The syntax is as follows:

```
provides <Name> version <Version>
```

Example from package "easycron":

```
provides cron version 3.10.0
```

The version number should be incemented by the OPT-developer in the third component, if only functional enhancements have been made and the OPT's interface is still. The version number should be increased in the first or second component when the interface has changed in any incompatible way (eg. due to variable renaming, path changes, missing or renamed utilities, etc.).

Definition of a Dependency to a Service with a specific Version: `depends`

If another service is needed to provide the own function (eg. a web server) this dependency to a specific version may be defined here. The version can be given with two (i.e. "2.1") or three numbers (i.e. "2.1.11") while the two-number version accepts all versions starting with this number and the three-number version only accepting just the specified one. A list of version numbers may also be specified if multiple versions of the service are compatible with the package.

The syntax is as follows:

```
depends on <Name> version <Version>+
```

An example: Package "server" contains:

```
provides server version 1.0.1
```

A Package "client" with the following `depends`-instruction is given:[2]

```
depends on server version 1.0       # OK, '1.0' matches '1.0.1'
depends on server version 1.0.1     # OK, '1.0.1' matches '1.0.1'
depends on server version 1.0.2     # Error, '1.0.2' does not match with '1.0.1'
depends on server version 1.1       # Error, '1.1' does not match with '1.0.1'
depends on server version 1.0 1.1   # OK, '1.0' matches '1.0.1'
depends on server version 1.0.2 1.1 # Error, neither '1.0.2' nor '1.1' are matching
                                    # '1.0.1'
```

[2] Of course only one at a time!

Communication with the User: `warning`, `error`, `fatal_error`

Using these three functions users may be warned, signalized an errors or stop the test immediately. The syntax is as follows:

- `warning "text"`
- `error "text"`
- `fatal_error "text"`

All strings passed to these funtions are subject of variable substitution (Page 298).

Assignments

If for any reason a temporary variable is required it can be created by "`set var [= value]`". *The variable can not be a configuration variable!* [3] If you omit the "= value" part the variable is simply set to "yes" so it may be tested in an `if`-statement. If an assignment part is given, anything may be specified after the equal sign: normal variables, indexed variables, numbers, strings and version numbers.

Please note that by the assignment also the *type* of the temporary variable is defined. If a number is assigned `mkfli4l` "remembers" that the variable contains a number and later on allows calculations with it. Trying to do calculations with variables of other types will fail. Example:

```
set i=1   # OK, i is a numeric variable
set j=i+1 # OK, j is a numeric variable and contains the value 2
set i="1" # OK, i now is a string variable
set j=i+1 # Error "You can't use a non-numeric ID in a numeric
          #         context. Check type of operand."
          # --> no calculations with strings!
```

You may also define temporary arrays (see below). Example:

```
set prim_%[1]=2
set prim_%[2]=3
set prim_%[3]=5
warning "${prim_n}"
```

The number of array elements is kept by `mkfli4l` in the variable `prim_n`. The code above hence leads to the following output:

```
Warning: 3
```

If the right side of an assignment is a string constant, it is subject of variable substitution (Page 298) at the time of assignment. The following example demonstrates this. The code:

[3] This is a desired restriction: Check scripts are *not* able to change the user configuration.

```
set s="a"
set v1="$s" # v1="a"
set s="b"
set v2="$s" # v2="b"
if (v1 == v2)
then
  warning "equal"
else
  warning "not equal"
fi
```

will output "not equal", because the variables `v1` and `v2` replace the content of the variable `s` already at the time of assignment.

Hint: A variable set in a script is visible while processing further scripts – currently there exists no such thing as local variables. Since the order of processing scripts of different packages is not defined, you should never rely on any variable having values defined by another package.

Arrays

If you want to access elements of a %-variable (of an array) you have to use the original name of the variable like mentioned in the file `check/<PACKAGE>.txt` and add an index for each "%" sign by using "[*Index*]".

Example: If you want to access the elements of variable `PF_USR_CHAIN_%_RULE_%` you need two indices because the variable has two "%" signs. All elements may be printed for example using the following code (the `foreach`-loop is exlained in see below (Page 307)):

```
foreach i in pf_usr_chain_n
do
    # only one index needed, only one '%' in the variable's name
    set j_n=pf_usr_chain_%_rule_n[i]
    # Attention: a
    # foreach j in pf_usr_chain_%_rule_n[i]
    # is not possible, hence the use of j_n!
    foreach j in j_n
    do
        # two indices needed, two '%' in the variable's name
        set rule=pf_usr_chain_%_rule_%[i][j]
        warning "Rule $i/$j: ${rule}"
    done
done
```

With this sample configuration

```
PF_USR_CHAIN_N='2'
PF_USR_CHAIN_1_NAME='usr-chain_a'
PF_USR_CHAIN_1_RULE_N='2'
PF_USR_CHAIN_1_RULE_1='ACCEPT'
PF_USR_CHAIN_1_RULE_2='REJECT'
PF_USR_CHAIN_2_NAME='usr-chain_b'
PF_USR_CHAIN_2_RULE_N='1'
PF_USR_CHAIN_2_RULE_1='DROP'
```

the following output is printed:

```
Warning: Rule 1/1: ACCEPT
Warning: Rule 1/2: REJECT
Warning: Rule 2/1: DROP
```

Alternatively, you can iterate directly over all values of the array (but the exact indices of the entries are not always known, because this is not required):

```
foreach rule in pf_usr_chain_%_rule_%
do
    warning "Rule %{rule}='${rule}'"
done
```

That produces the following output with the sample configuration from above:

```
Warning: Rule PF_USR_CHAIN_1_RULE_1='ACCEPT'
Warning: Rule PF_USR_CHAIN_1_RULE_2='REJECT'
Warning: Rule PF_USR_CHAIN_2_RULE_1='DROP'
```

The second example nicely shows the meaning of the `%<Name>`-syntax: Within the string `%rule` is substitued by the *name* of the variable in question (for example `PF_USR_CHAIN_1_RULE_1`), while `$rule` is substituted by its *content* (i.e. `ACCEPT`).

Encryption of Passwords: `crypt`

Some variables contain passswords that should not be noted in plain text in `rc.cfg`. These variables can be encrypted by the use of `crypt` and are transferred to a format also needed on the router. Use this like here:

```
crypt (<Variable>)
```

The `crypt` function is the *only* point at which a configuration variable can be changed.

Querying File Properties: `stat`

`stat` is used to query file properties. At the moment only file size can be accessed. If checking for files under the current configuration directory you may use the internal variable `config_dir`. The Syntax:

```
stat (<file name>, <key>)
```

The command looks like this (the parameters used are only examples):

```
foreach i in openvpn_%_secret
do
   stat("${config_dir}/etc/openvpn/$i.secret", keyfile)
   if (keyfile_res != "OK")
   then
      error "OpenVPN: missing secretfile <config>/etc/openvpn/$i.secret"
   fi
done
```

The example checks whether a file exists in the current configuration directory. If `OPENVPN_1_SECRET='test'` is set in the configuration file, the loop in the first run checks for the existence of the file `etc/openvpn/test.secret` in the current configuration directory.

After the call two variables are defined:

- `<Key>_res`: Result of the system call stat() ("OK", if system call was successful, else the error message of the system call)
- `<Key>_size`: File size

It may for example look like this:

```
stat ("unix/Makefile", test)
if ("$test_res" == "OK")
then
        warning "test_size = $test_size"
else
        error "Error '$test_res' while trying to get size of 'unix/Makefile'"
fi
```

A file name passed as a string constant is subject of variable substitution (Page 298).

Search files: `fgrep`

If you wish to search a file via "grep"[4] you may use the `fgrep` command. The syntax is:

```
fgrep (<File name>, <RegEx>)
```

If the file `<File name>` does not exist `mkfli4l` will abort with a fatal error! If it is not sure if the file exists, test this before with `stat`. After calling `fgrep` the search result is present in an array called `FGREP_MATCH_%`, with its index *x* as usual ranging from one to `FGREP_MATCH_N`. `FGREP_MATCH_1` points to the whole range of the line the regular expression has matched, while `FGREP_MATCH_2` to `FGREP_MATCH_N` contain the *n-1* th part in brackets.

A first example will illustrate the use. The file `opt/etc/shells` contains the line:

```
/bin/sh
```

The following code

```
fgrep("opt/etc/shells", "^/(.)(.*)/")
foreach v in FGREP_MATCH_%
do
  warning "%v='$v'"
done
```

produces this output:

```
Warning: FGREP_MATCH_1='/bin/'
Warning: FGREP_MATCH_2='b'
Warning: FGREP_MATCH_3='in'
```

[4] "grep" is a common command on *nix-like OSes for filtering text streams.

The RegEx has (only) matched with "/bin/" (only this part of the line is contained in the variable `FGREP_MATCH_1`). The first bracketed part in the expression only matches the first char after the first "/", this is why only "b" is contained in `FGREP_MATCH_2`. The second bracketed part contains the rest after "b" up to the last "/", hence "in" is noted in variable `FGREP_MATCH_3`.

The following second example demonstrates an usual use of `fgrep` taken from `check/base.ext`. It will be tested if all `tmpl:`-references given in `PF_FORWARD_x` are really present.

```
foreach n in pf_forward_n
do
  set rule=pf_forward_%[n]
  if (rule =~ "tmpl:([^[:space:]]+)")
  then
    foreach m in match_%
    do
      stat("$config_dir/etc/fwrules.tmpl/$m", tmplfile)
      if(tmplfile_res == "OK")
      then
        add_to_opt "etc/fwrules.tmpl/$m"
      else
        stat("opt/etc/fwrules.tmpl/$m", tmplfile)
        if(tmplfile_res == "OK")
        then
          add_to_opt "etc/fwrules.tmpl/$m"
        else
          fgrep("opt/etc/fwrules.tmpl/templates", "^$m[[:space:]]+")
          if (fgrep_match_n == 0)
          then
            error "Can't find tmpl:$m for PF_FORWARD_${n}='$rule'!"
          fi
        fi
      fi
    done
  fi
done
```

Both a filename value as well as a regular expression passed as a string constant are subject to variable substitution (Page 298).

Splitting Parameters: `split`

Often variables can be assigned with several parameters, which then have to be split apart again in the startup scripts. If it is desired to split these previously and perform tests on them `split` can be used. The syntax is like this:

```
split (<String>, <Array>, <Separator>)
```

The string can be specified by a variable or directly as a constant. `mkfli4l` splits it where a separator is found and generates an element of the array for each part. You may iterate over these elements later on and perform tests. If nothing is found between two separators an array element with an empty string as its value is created. The exception is " ": Here all spaces are deleted and no empty variable is created.

If the elements generated by such a split should be in a numeric context (e.g. as indices) this has to be specified when calling `split`. This is done by the additional attribute "numeric". Such a call looks as follows:

```
split (<String>, <Array>, <Separator>, numeric)
```

An example:

```
set bar="1.2.3.4"
split (bar, tmp_%, '.', numeric)
foreach i in tmp_%
do
        warning "%i = $i"
done
```

the output looks like this:

```
Warning: TMP_1 = 1
Warning: TMP_2 = 2
Warning: TMP_3 = 3
Warning: TMP_4 = 4
```

Hint: If using the "numeric" variant `mkfli4l` will *not* check the generated string parts for really being numeric! If you use such a non-numeric construct later in a numeric context (i.e. in an addtion) `mkfli4l` will raise a fatal error. Example:

```
set bar="a.b.c.d"
split (bar, tmp_%, '.', numeric)
# Error: invalid number 'a'
set i=tmp_%[1]+1
```

A string constant passed to `split` in the first parameter is subject of variable substitution (Page 298).

Adding Files to the Archives: `add_to_opt`

The function `add_to_opt` can add additional files to the Opt- or RootFS-Archives. *All* files under `opt/` or from the configuration directory may be chosen. There is no limitation to only files from a specific package. If a file is found under `opt/` as well as in the configuration directory, `add_to_opt` will prefer the latter. The function `add_to_opt` is typically used if complex logical rules decide if and what files have to be included in the archives.

The syntax looks like this:

```
add_to_opt <File> [<Flags>]
```

Flags are optional. The defaults from table 8.2 are used if no flags are given.
See an example from the package "sshd":

```
if (opt_sshd)
then
   foreach pkf in sshd_public_keyfile_%
   do
     stat("$config_dir/etc/ssh/$pkf", publickeyfile)
     if(publickeyfile_res == "OK")
     then
         add_to_opt "etc/ssh/$pkf" "mode=400 flags=utxt"
     else
         error "sshd: missing public keyfile %pkf=$pkf"
     fi
   done
fi
```

`stat` (Page 303) at first checks for the file existing in the configuration directory. If it is, it will be included in the archive, if not, `mkfli4l` will abort with an error message.

Hint: Also for `add_to_opt` `mkfli4l` will first check (Page 289) if the file to be copied can be found in the configuration directory.

Filenames as well flags passed as string constants are subject of variable substitution (Page 298).

Control Flow

```
if (expr)
then
        statement
else
        statement
fi
```

A classic case distinction, as we know it. If the condition is true, the `then` part is executed, if the condition is wrong the `else` part.

If you want to run tests on array variables, you have to test every single variable. The `foreach` loop in two variants for this.

1. Iterate over array variables:

   ```
   foreach <control variable> in <array variable>
   do
           <instruction>
   done

   foreach <control variable> in <array variable-1> <array variable-2> ...
   do
           <instruction>
   done
   ```

   This loop iterates over all of the specified array variables, each starting with the first to the last element, the number of elements in this array is taken from the N-variable

associated with this array. The control variable takes the values of the respective array variables. It should be noted that when processing optional array variables that are not present in the configuration, an empty element is generated. You may have to take this into account in the script, for example like this:

```
foreach i in template_var_opt_%
do
    if (i != "")
    then
        warning "%i is present (%i='$i')"
    else
        warning "%i is undefined (empty)"
    fi
done
```

As you also can see in the example, the *name* of the respective array variables can be determined with the `%<control variable>` construction.

The instruction in the loop may be one of the above control elements or functions (`if`, `foreach`, `provides`, `depends`, ...).

If you want to access exactly one element of an array, you can address it using the syntax `<Array>[<Index>]`. The index can be a normal variable, a numeric constant or again an indexed array.

2. Iteration over N-variables:

```
foreach <control variable> in <N-variable>
do
        <instruction>
done
```

This loop executes from 1 to the value that is given in the N-variable. You can use the control variable to index array variables. So if you want to iterate over not only one but more array variables at the same time all controlled by the *same* N-variable you take this variant of the loop and use the control variable for indexing multiple array variables. Example:

```
foreach i in host_n
do
    set name=host_%_name[i]
    set ip4=host_%_ip4[i]
    warning "$i: name=$name ip4=$ip4"
done
```

The resulting content of the `HOST_%_NAME`- and `HOST_%_IP4`-arrays for this example:

```
Warning: 1: name=berry ip4=192.168.11.226
Warning: 2: name=fence ip4=192.168.11.254
Warning: 3: name=sandbox ip4=192.168.12.254
```

Expressions

Expressions link values and operators to a new value. Such a value can be an normal variable, an array element, or a constant (Number, string or version number). All string constants in expressions are subject to variable substitution (Page 298).

Operators allow just about everything you could want from a programming language. A test for the equality of two variables could look like this:

```
var1 == var2
"$var1" == "$var2"
```

It should be noted that the comparison is done depending on the type that was defined for the variable in `check/<PACKAGE>.txt`. If one of the two variables is numeric (Page 298) the comparison is made numeric-based, meaning that the strings are converted to numbers and then compared. Otherwise, the comparison is done string-based; comparing `"05" == "5"` gives the result "false", a comparison `"18" < "9"` "true" due to the lexicographical string order: the digit "1" precedes the digit "9" in the ASCII character set.

For the comparison of version numbers the construct `numeric(version)` is introduced, which generates the numeric value of a version number for comparison purposes. Here applies:

```
numeric(version) := major * 10000 + minor * 1000 + sub
```

whereas "major" is the first component of the version number, "minor" the second and "sub" the third. If "sub" is missing the term in the addition above is omitted (in other words "sub" will be equalled to zero).

A complete list of all expressions can be found in table 8.3. "val" stands for any value of any type, "number" for a numeric value and "string"for a string.

Table 8.3.: Logical Epressions

Expression	true if
id	id == "yes"
val == val	values of identical type are equal
val != val	values of identical type are unequal
val == number	numeric value of val == number
val != number	numeric value of val != number
val < number	numeric value of val < number
val > number	numeric value of val > number
val == version	numeric(val) == numeric(version)
val < version	numeric(val) < numeric(version)
val > version	numeric(val) > numeric(version)
val =~ string	regular expression in string matches val
(expr)	Expression in brackets is true
expr && expr	both expressions are true
expr || expr	at least one of both expressions is true
copy_pending(id)	see description
samenet (string1, string2)	string1 describes the same net as string2
subnet (string1, string2)	string1 describes a subnet of string2

Match-Operator

With the match operator `=~` you can check whether a regular expression matches the value of a variable. Furthermore, one can also use the operator to extract subexpressions from a variable. After successfully applying a regular expression on a variable the array `MATCH_%` contains the parts found. May look like this:

```
set foo="foobar12"
if ( foo =~ "(foo)(bar)([0-9]*)" )
then
        foreach i in match_%
        do
                warning "match %i: $i"
        done
fi
```

Calling `mkfli4l` then would lead to this output:

```
Warning: match MATCH_1: foo
Warning: match MATCH_2: bar
Warning: match MATCH_3: 12
```

When using `=~` you may take all existing regular expressions into account. If you i.e. want to check whether a PCMCIA Ethernet driver is selected without `OPT_PCMCIA` being set to "yes", it might look like this:

```
if (!opt_pcmcia)
then
    foreach i in net_drv_%
    do
       if (i =~ "^(RE:PCMCIA_NET_DRV)$")
       then
           error "If you want to use ..."
       fi
    done
fi
```

As demonstrated in the example, it is important to *anchor* the regular expression with `^` and `$` if intending to apply the expression on the *complete* variable. Otherwise, the match-expression already returns "true" if only a *part* of the variable is covered by the regular expression, which is certainly not desired in this case.

Check if a File has been copied depending on the Value of a Variable: `copy_pending`

With the information gained during the checking process the function `copy_pending` tests if a file has been copied depending on the value of a variable or not. This can be used i.e. in order to test whether the driver specified by the user really exists and has been copied. `copy_pending` accepts the name to be tested in the form of a variable or a string. [5] In order to accomplish this `copy_pending` checks whether

[5] As described before the string is subject of variable substitution, i.e via a `foreach`-loop (Page 307) and a `%<Name>`-subsitution (Page 298) all elements of an array may be examined.

- the variable is active (if it depends on an OPT it has to be set to “yes”),
- the variable was referenced in an `opt/<PACKAGE>.txt`-file and
- whether a file was copied dependant on the current value.

`copy_pending` will return “true” if it detects that during the last step *no* file was copied, the copy process hence still is “pending”.

A small example of the use of all these functions can be found in `check/base.ext`:

```
foreach i in net_drv_%
do
    if (copy_pending("%i"))
    then
        error "No network driver found for %i='$i', check config/base.txt"
    fi
done
```

Alle elements of the array `NET_DRV_%` are detected for which no copy action has been done because there is no corresponding entry existing in `opt/base.txt`.

Comparison of Network Addresses: `samenet` und `subnet`

For testing routes from time to time a test is needed whether two networks are identical or if one is a subnet of the other. The two functions `samenet` and `subnet` are of help here.

```
samenet (netz1, netz2)
```

returns “true” if both nets are identical and

```
subnet (net1, net2)
```

returns “true” if “net1” is a subnet of “net2”.

Expanding the Kernel Command Line

If an OPT must pass other boot parameters to the kernel, in former times the variable `KERNEL_BOOT_OPTION` had to be checked whether the required parameter was included, and if necessary, a warning or error message had to be displayed. With the internal variable `KERNEL_BOOT_OPTION_EXT` you may add a necessary but missing option directly in an ext-script. An Example taken from `check/base.ext`:

```
if (powermanagement =~ "apm.*|none")
then
    if ( ! kernel_boot_option =~ "acpi=off")
    then
        set kernel_boot_option_ext="${kernel_boot_option_ext} acpi=off"
    fi
fi
```

This passes “acpi=off” to the kernel if no or “APM”-type power management is desired.

8.3.8. Support for Different Kernel Version Lines

Different kernel version lines often differ in some details:

- changed drivers are provided, some are deleted, others are added
- module names simply differ
- module dependencies are different
- modules are stored in different locations

These differences are mostly handled automatically by `mkfli4l`. To describe the available modules you can, on one hand expand tests dependant on the version (conditional regular expressions (Page 294)), or, on the other hand `mkfli4l` allows *version dependant* `opt/<PACKAGE>.txt`-files. These are then named `opt/<PACKAGE>_<Kernel-Version>.txt`, where the components of the kernel version are separated from each other by underscores. An example: the package "base" contains these files in its `opt`-directory:

- `base.txt`
- `base_3_14.txt`
- `base_3_17.txt`

the first file (`base.txt`) is *always* considered. Both other files are only considered if the kernel version is called "3.14(.*)" resp. "3.17(.*)". As seen here, some parts of the version may be omitted in file names, if a group of kernels should be addressed. If `KERNEL_VERSION='3.14.26'` is given, the following files (if existing) are considered for the package `<PACKAGE>`:

- `<PACKAGE>.txt`
- `<PACKAGE>_3.txt`
- `<PACKAGE>_3_14.txt`
- `<PACKAGE>_3_14_26.txt`

8.3.9. Documentation

Documentation should be placed in the files

- `doc/<LANGUAGE>/opt/<PACKAGE>.txt`
- `doc/<LANGUAGE>/opt/<PACKAGE>.html`.

HTML-files may be splitted, meaning one for each OPT contained. Nevertheless a file `<PACKAGE>.html` has to be created linking to the other files. Changes should be documented in:

- `changes/<PACKAGE>.txt`

The entire text documentation may not contain any tabs and has to have a line feed no later than after 79 characters. This ensures that the documentation can also be read correctly with an editor without automatic line feed.

Also a documentation in LaTeX-format is possible, with HTML and PDF versions generated from it. The documentation of fli4l may serve as an example here. A documentation framework for required LaTeX-macros can be found in the package "template". A brief description is to be found in the following subsections.

The fli4l documentation is currently available in the following languages: German (`<LANGUAGE>` = "deutsch"), English (`<LANGUAGE>` = "english") and French (`<LANGUAGE>` = "french"). It is the package developer's decision to document his package in any language. For the purposes of clarity it is recommended to create a documentation in German and/or English (ideally in both languages).

Prerequisites for Creating a LaTeXDocumentation

To create a documentation from LaTeX-sources the following requirements apply:

- Linux/OS X-Environment: For ease of production, a makefile exists to automate all other calls (Cygwin should work too, but is not tested by the fli4l team)
- LaTeX2HTML for the HTML version
- of course LaTeX (Recommended: "TeX Live" for Linux/OS X and "MiKTeX" for Microsoft Windows) the "pdftex"program and these TeX-packages:
  - current KOMA-Skript (at least version 2)
  - all packages necessary for pdftex
  - unpacked documentation package for fli4l, it provides the necessary makefiles and TeX-styles

File Names

The documentation files are named according to the following scheme:

`<PACKAGE>_main.tex`: This file contains the main part of the documentation. `<PACKAGE>` stands for the name of the package to be described (in lowercase letters).

`<PACKAGE>_appendix.tex`: If further comments should be added to the package, they should be placed there.

The files should be stored in the directory `fli4l/<PACKAGE>/doc/<SPRACHE>/tex/<PACKAGE>`. For the package sshd this looks like here:

```
$ ls fli4l/doc/deutsch/tex/sshd/
Makefile sshd_appendix.tex  sshd_main.tex  sshd.tex
```

The Makefile is responsible for generating the documentation, the `sshd.tex`-file provides a framework for the actual documentation and the appendix, which is located in the other two files. See an example in the documentation of the package "template".

LaTeX-Basics

LaTeX is, just like HTML, "Tag-based" , only that the tags are called "commands" and have this format: `\command` resp. `\begin{environment}` ... `\end{environment}`

By the help of commands you should rather emphasize the *importance* of the text less the *display*. It is therefore of advantage to use

`\warning{Please␣do␣not...}`

instead of

`\emph{Please␣do␣not...}`

.

Each command rsp. each environment may take some more parameters noted like this: `\command{parameter1}{parameter2}{parameterN}`.

Some commands have optional parameters in square (instead of curly) brackets: `\command[optionalParameter]{parameter1}` ... Usually only one optional parameter is used, in rare cases there may be more.

Individual paragraphs in the document are separated by blank lines. Within these paragraphs LaTeX itself takes care of line breaks and hyphenation.

The following characters have special meaning in LaTeX and, if occuring in normal text, must be masked prefixed by a \: # $ & _ % { }. "~" and "^" have to be written as follows: `\verb?~?` `\verb?^?`

The main LaTeX-commands are explained in the documentation of the package "template".

8.3.10. File Formats

All text files (both documentation and scripts, which later reside on the router) should be added to the package in DOS file format, with CR/LF instead of just LF at the end of a line. This ensures that Windows users can read the documentation even with "notepad" and that after changing a script under Windows everything still is executable on the router.

The scripts are converted to the required format during archive creation (see the description of the flags in table 8.2).

8.3.11. Developer Documentation

If a program from the package defines a new interface that other programs can use, please store the documentation for this interface in a separate documentation in `doc/dev/<PACKAGE>.txt`.

8.3.12. Client Programs

If a package also provides additional client programs, please store them in the directory `windows/` for Windows clients and in the directory `unix/` for *nix and Linux clients.

8.3.13. Source Code

Customized programs and source code may be enclosed in the directory `src/<PACKAGE>/`. If the programs should be built like the rest of the fi4l programs, please have a look at the documentation of the "src"-package (Page **??**) .

8.3.14. More Files

All files, which will be copied to the router have to be stored under `opt/etc/` and `opt/files/`. Be under

- `opt/etc/boot.d/` and `opt/etc/rc.d/`: scripts, that should be executed on system start
- `opt/etc/rc0.d/`: scripts, that should be executed on system shutdown
- `opt/etc/ppp/`: scripts, that should be executed on dialin or hangup
- `opt/files/`: executable programs and other files according to their positions in the file system (for example the file `opt/files/bin/busybox` will later be situated in the directory `/bin` on the router)

Scripts in `opt/etc/boot.d/`, `opt/etc/rc.d/` and `opt/etc/rc0.d/` have the following naming scheme:

```
rc<number>.<name>
```

The number defines the order of execution, the name gives a hint on what program/package is processed by this script.

8.4. Creating Scripts for fli4l

The following is *not* a general introduction to shell scripts, everyone can read about this topic on the Internet. It is only about the flil4-specific things. Further information is available in the various *nix/Linux help pages. The following links may be used as entry points to this topic:

- Introduction to shell scripts:
  - http://linuxcommand.org/writing_shell_scripts.php
- Help pages online:
  - http://linux.die.net/
  - http://heapsort.de/man2web
  - http://man.he.net/
  - http://www.linuxcommand.org/superman_pages.php

8.4.1. Structure

In the *nix world, it is necessary to begin a script with the name of the interpreter, hence the first line is:

```
#!/bin/sh
```

To easily recognize later what a script does and who created it, this line should now be followed by a short header, like so:

```
#------------------------------------------------------------------
# /etc/rc.d/rc500.dummy - start my cool dummy server
#
# Creation:     19.07.2001 Sheldon Cooper  <sheldon@nerd.net>
# Last Update:  11.11.2001 Howard Wolowitz <howard@nerd.net>
#------------------------------------------------------------------
```

Now for the real stuff to start...

8.4.2. Handling of Configuration Variables

Packages are configured via the file `config/<PACKAGE>.txt`. The active variables (Page 289) contained there are transferred to the file `rc.cfg` during creation of the medium. This file is processed during router boot before any rc-script (scripts under `/etc/rc.d/`) gets started. The script then may access all configuration variables by reading the content of `$<variable name>`.

If the values of configuration variables are needed after booting the may be extracted from `/etc/rc.cfg` to which the configuration of the boot medium was written during the boot process. If for example the value of the variable `OPT_DNS` should be processed in a script this can be achieved as follows:

```
eval $(grep "^OPT_DNS=" /etc/rc.cfg)
```

This works efficiently also with multiple variables (with calling the `grep` program only once):

```
eval $(grep "^\(HOSTNAME\|DOMAIN_NAME\|OPT_DNS\|DNS_LISTEN_N\)=" /etc/rc.cfg)
```

8.4.3. Persistent Data Storage

Occasionally a package needs the possibility to store data persistent, surviving the reboot of the router. For this purpose, the function `map2persistent` exists and can be called from a script in `/etc/rc.d/`. It expects a variable that contains a path and a subdirectory. The idea is that the variable is either describing an actual path – then this path is used because the user explicitely has done so, or the string “auto” – then a subdirectory on a persistent medium is created corresponding to the second parameter. The function returns the result in the variable passed by name as the first parameter. An example will make this clear. `VBOX_SPOOLPATH` is a variable, that ’contains a path or the string “auto” . The call

```
begin_script VBOX "Configuring vbox ..."
[...]
map2persistent VBOX_SPOOLPATH /spool
[...]
end_script
```

results in the variable `VBOX_SPOOLPATH` either not being changed at all (if it contains a path), or being changed to `/var/lib/persistent/vbox/spool` (if it contains the string “auto”). `/var/lib/persistent` then points[6] to a directory on a non-volatile and writable storage medium, `<SCRIPT>` is the name of the calling script in lowercase (this name is derived from

[6] by the use of a so-called “bind”-mount

the first argument of the begin_script-call (Page 317)). If no suitable medium should exist (which may well be), /var/lib/persistent is a directory in the RAM disk.

Please note that the path returned by map2persistent is *not* created automatically – The caller has to do that by himself (ie. by calling mkdir -p <path>).

The file /var/run/persistent.conf allows for checking if persistent data storage is possible. Example:

```
. /var/run/persistent.conf
case $SAVETYPE in
persistent)
    echo "persistent data storage is possible!"
    ;;
transient)
    echo "persistent data storage is NOT possible!"
    ;;
esac
```

8.4.4. Debugging

For startup scripts it is often useful to run them in debug mode in a shell when you are in need to determine where they fail. For this purpose, insert the following at the beginning and at the end:

```
begin_script <OPT-Name> "start message"
<script code>
end_script
```

At the start and at the end of the script the specified text will now appear, preceded by "finished".

If you want to debug the scripts, you must do two things:

1. You have to set DEBUG_STARTUP (Page 30) to "yes".
2. You have to activate debugging for the OPT. This is usually done by the entry

   ```
   <OPT-Name>_DO_DEBUG='yes'
   ```

   in the config file.[7] What happens during runtime is now displayed in detail on screen.

Further Variables helpful for Debugging

DEBUG_ENABLE_CORE This variable allows the creation of "Core-Dumps". If a program crashes due to an error an image of the current state in the file system is stored which can be used to analyse the problem. The core dumps are stored under /var/log/dumps/.

DEBUG_IP Activating this variable will log all calls of the program ip.

DEBUG_IPUP Setting this variable to "yes" will log all executed instructions of the ip-up- and ip-down-scripts to syslog.

[7] Sometimes multiple start-scripts are used, which then have different names for their debug-variables. Have a quick look at the scripts for clarification.

LOG_BOOT_SEQ Setting this variable to "yes" will cause `bootlogd` to log all console output during boot to the file `/var/tmp/boot.log`. This variable has "yes" as a default value.

DEBUG_KEEP_BOOTLOGD Normally `bootlogd` is terminated at the end of the boot process. Activating this variable prevents this and thus allows for logging console output during the whole runtime.

DEBUG_MDEV Setting this variable generates a logfile for the `mdev`-daemon, which is responsible for creating device nodes under `/dev`.

8.4.5. Hints

- It is *always* better to use curly brackets "{...}" instead of normal ones"(...)". However, care must be taken to ensure that after the opening bracket a space or a new line follows before the next command and before the closing brackets a Semicolon or a new line. For example:

```
{ echo "cpu"; echo "quit"; } | ...
```

  is equal to:

```
{
        echo "cpu"
        echo "quit"
} | ...
```

- A script may be stopped by an "exit" at any time. This is deadly for start scripts (`opt/etc/boot.d/...`, `opt/etc/rc.d/...`), stop-scripts (`opt/etc/rc0.d/...`) and `ip-up`/`ip-down`-scripts (`opt/etc/ppp/...`) because the following scripts will not be run as well. If in doubt, keep your fingers away.

- KISS – Keep it small and simple. You want to use Perl for scripting? The scripting abilities of fli4l are not enough for you? Rethink your attitude! Is your OPT really necessary? fli4l after all is "only" a router and a router should not really offer server services.

- The error message ": not found" usually means the script is still in DOS format. Another source of errors: the script is not executable. In both cases the `opt/<PACKAGE>.txt` file should be checked whether it contains the correct options for "mode", "gid", "uid" and Flags. If the script is created during boot, execute "chmod +x <script name>".

- Use the path `/tmp` for temporary files. However, it is important to keep in mind that there is little space there because it is situated in the rootfs-RAM disc! If more space is needed, you have to create and mount your own ramdisk. Detailed informations on this topic can be found in the section "RAM-Disks" of this documentation.

- In order to create temporary files with unique names you should always append the current process-ID stored in the shell variable "$" to the file name. `/tmp/<OPT-Name>.$$` hence is a perfect file name, `/tmp/<OPT-Name>` rather not (`<OPT-Name>` of course has to be replaced by the according OPT-Name).

8.5. Using The Packet Filter

8.5.1. Adding Own Chains And Rules

A set of routines is provided to manipulate the packet filter to add or delete so-called "chains" and "rules". A chain is a named list of ordered rules. There is a set of predefined chains (`PREROUTING`, `INPUT`, `FORWARD`, `OUTPUT`, `POSTROUTING`), using this set of routines more chains can be created as needed.

`add_chain/add_nat_chain <chain>`: Adds a chain to the "filter-" or "nat-" table.

`flush_chain/flush_nat_chain <chain>`: Deletes all rules from a chain of the "filter-" or "nat-" table.

`del_chain/del_nat_chain <chain>`: Deletes a chain from the "filter-" or "nat-" table. Chains must be empty prior to deleting and all references to them have to be deleted as well before. Such a reference i.e. can be a JUMP-action with the chain defined as its target.

`add_rule/ins_rule/del_rule`: Adds rules to the end (`add_rule`) resp. at any place of a chain (`ins_rule`) or deletes rules from a chain (`del_rule`). Use the syntax like here:

```
add_rule <table> <chain> <rule> <comment>
ins_rule <table> <chain> <rule> <position> <comment>
del_rule <table> <chain> <rule> <comment>
```

where the parameters have the following meaning:

table The table in which the chain is

chain The chain, in which the rule is to be inserted

rule The rule which is to be inserted, the format corresponds to that used in the configuration file

position The position at which the rule will be added (only in `ins_rule`)

comment A comment that appears with the rule when somebody looks at the packet filter.

8.5.2. Integrating Into Existing Rules

fli4l configures the packet filter with a certain default rule set. If you want to add your own rules, you will usually want to insert them after the default rule set. You will also need to know what the action is desired by the user when dropping a packet. This information can be obtained for `FORWARD`- and `INPUT` chains by calling two functions, `get_defaults` and `get_count`. After calling

```
get_defaults <chain>
```

the following results are obtained:

`drop`: This variable contains the chain to which is branched when a packet is discarded.

reject: This variable contains the chain to which is branched when a packet is rejected.

After calling

```
get_count <chain>
```

the variable res contains the number of rules in the chain <chain>. This position is of importance because you can *not* simply use add_rule to add a rule at the end of the predefined "filter"-chains INPUT, FORWARD and OUTPUT. This is because these chains are completed with a default rule valid for all remaining packets depending on the content of the PF_<chain>_POLICY-variable. Adding a rule *after* this last rule hence has no effect. The function get_count instead allows to detect the position right *in front of* this last rule and to pass this position to the ins_rule-function as a parameter <position> in order to add the rule in the place at the end of the appropriate chain, but right in front of this last default rule targeting all remaining packets.

An example from the script opt/etc/rc.d/rc390.dns_dhcp from the package "dns_dhcp" shall make this clear:

```
case $OPT_DHCPRELAY in
    yes)
        begin_script DHCRELAY "starting dhcprelay ..."

        idx=1
        interfaces=""
        while [ $idx -le $DHCPRELAY_IF_N ]
        do
            eval iface='$DHCPRELAY_IF_'$idx

            get_count INPUT
            ins_rule filter INPUT "prot:udp  if:$iface:any 68 67 ACCEPT" \
                $res "dhcprelay access"

            interfaces=$interfaces' -i '$iface
            idx=`expr $idx + 1`
        done
        dhcrelay $interfaces $DHCPRELAY_SERVER

        end_script
    ;;
esac
```

Here you can see in the middle of the loop a call to get_count followed by a call to the ins_rule function and, among other things, the res variable is passed as position parameter.

8.5.3. Extending The Packet Filter Tests

fli4l uses the syntax match:params in packet filter rules to add additional conditions for packet matching (see mac:, limit:, length:, prot:, ...). If you want to add tests you have to do this as follows:

1. Creating a file opt/etc/rc.d/fwrules-<name>.ext. The content of this file is something like this:

```
# extension is available
foo_p=yes

# the actual extension, adding matches to match_opt
do_foo()
{
    param=$1
    get_negation $param
    match_opt="$match_opt -m foo $neg_opt --fooval $param"
}
```

2. Testing the extension:

```
$ cd opt/etc/rc.d
$ sh test-rules.sh 'foo:bar ACCEPT'
add_rule filter FORWARD 'foo:bar ACCEPT'
iptables -t filter -A FORWARD -m foo --fooval bar -s 0.0.0.0/0 \
    -d 0.0.0.0/0 -m comment --comment foo:bar ACCEPT -j ACCEPT
```

3. Adding the extension and all other needed files (`iptables` components) to the archive using the known mechanisms.

4. Allowing the extension in the configuration by extending of `FW_GENERIC_MATCH` in an exp-file, for example:

```
+FW_GENERIC_MATCH(OPT_FOO) = 'foo:bar' : ''
```

8.6. CGI-Creation for Package *httpd*

8.6.1. General information about the web server

The web server used in fli4l is `mini_httpd` by ACME Labs. The sources can be found at http://www.acme.com/software/mini_httpd/. However, a few changes in the current version were made for fli4l. The modifications are located in the *src* package in the directory `src/fbr/buildroot/package/mini_httpd`.

8.6.2. Script Names

The script names should be self-explanatory in order to be easy to distinguish from other scripts and even similar names have to be avoided to differ from other OPTs.

To make scripts executable and convert DOS line breaks to UNIX ones a corresponding entry has to be created in `opt/<PACKAGE>.txt`, see Table 8.2 (Page 287).

8.6.3. Menu Entries

To create an entry in the menu you have to enter it in the file `/etc/httpd/menu`. This mechanism enables OPTs to change the menu during runtime. This should only be done using the script `/etc/httpd/menu` because this will check for valid file formatting. New menu items are inserted as follows:

```
httpd-menu.sh add [-p <priority>] <link> <name> [section] [realm]
```

Thus, an entry with the name `<name>` is inserted to the `[section]`. If `[section]` is omitted, it will be inserted in the section "OPT-Packages" as default. `<link>` specifies the target of the new link. `<priority>` specifies the priority of a menu item in its section. If not set, the default priority used is 500. The priority should be a three digit number. The lower the priority, the higher the link is placed in the section. If an entry should be placed as far down as possible the priority to choose is e.g. 900. Entries with the same priority are sorted by the target of the link. In `[realm]` the range is specified for which a logged-in user must have `view` rights so the item is displayed for him. If `[realm]` is not specified, the menu item is always displayed. For this, see also the section "User access rights" (Page 326).

Example:

```
httpd-menu.sh add "newfile.cgi" "Click here" "Tools" "tools"
```

This example creates a link named "Click here" with the target "newfile.cgi" in the section "Tools" which will be created if not present.

The script may also delete entries from a menu:

```
httpd-menu.sh rem <link>
```

By executing this the entry containing the link `<link>` will be deleted.

Important: *If several entries have the same link target file they will all be removed from the menu.*

Since sections can have priorities they can also be created manually. If a section was created automatically when adding a menu entry it defaults to priority 500. The syntax for creating sections is as follows:

```
httpd-menu.sh addsec <priority> <name>
```

`<priority>` should only be a three digit number.

In order to create meaningful priorities it is worthwhile to have a look at the file `/etc/httpd/menu` of fli4l during runtime, priorities are placed in the second column.

A short description of the file format of the file menu follows for completeness. Those satisfied with the function of `httpd-menu.sh` may skip this section. The file `/etc/httpd/menu` is divided into four columns. The first column is a letter identifying the line as a heading or a menu entry. The second column is the sort priority. The third column contains the target of the link for entries and for headlines a hyphen, as this field has no meaning for headings. The rest of the line is the text that will appear in the menu.

Headings use the letter "t", a new menu section will be started then. Normal menu items use the letters "e". An example:

```
t 300 - My beautiful OPT
e 200 myopt1.cgi Do something beautiful
e 500 myopt1.cgi?more=yes Do something even more beautiful
```

When editing this file you have to be aware that the script `httpd-menu.sh` always stores the file sorted. The individual sections are sorted and the entries in this section are sorted too. The sorting algorithm can be stolen from `httpd-menu.sh`, however, it would be better to expand the script itself with possible new functions, so that all menu-editing takes place at a central location.

8.6.4. Construction of a CGI script

The headers

All web server scripts are simple shell scripts (interpreter as e.g. Perl, PHP, etc. are much too big in filesize for fli4l). You should start with the mandatory script header (reference to the interpreter, name, what does the script, author, license).

Helper Script `cgi-helper`

After the header you should include the helper script `cgi-helper` with the following call:

```
. /srv/www/include/cgi-helper
```

A space between the dot and the slash is important!

This script provides several helper functions that should greatly simplify the creation of CGIs for fli4l. With the integration some standard tasks are performed, such as the parsing of variables that were passed via forms or via the URL or loading of language and CSS files.

Here is a small function overview:

Table 8.4.: Functions of the `cgi-helper` script

Name	Function
`check_rights`	Check for user access rights
`http_header`	Creation of a standard HTTP header or a special header, e.g. for file download
`show_html_header`	Creation of a complete page header (inc. HTTP header, headline and menu)
`show_html_footer`	Creation of a footer for the HTML page
`show_tab_header`	Creation of a content window with tabs
`show_tab_footer`	Creation of a footer for the content window
`show_error`	Creation of a box for error messages (background color: red)
`show_warn`	Creation of a box for warning messages (background color: yellow)
`show_info`	Creation of a box for information/ success messages (background color: green)

Contents of a CGI script

To ensure consistency in design and especially the compatibility with future versions of fli4l it is highly recommended to use the functions of the `cgi-helper` script even if theoretically everything in a CGI could be generated from scratch.

A simple CGI script might look like this:

```
#!/bin/sh
# --------------------
# Header (c) Author Date
# --------------------
# get main helper functions
. /srv/www/include/cgi-helper

show_html_header "My first CGI"
echo '   <h2>Welcome</h2>'
echo '   <h3>This is a CGI script example</h3>'
show_html_footer
```

The Function `show_html_header`

The `show_html_header` function expects a string as a parameter. This string represents the title of the generated page. It automatically generates the menu and includes associated CSS and language files as long as they can be found in the directories `/srv/www/css` resp. `/srv/www/lang` and have the same name (but of course a different extension) as the script. An example:

```
/srv/www/admin/OpenVPN.cgi
/srv/www/css/OpenVPN.css
/srv/www/lang/OpenVPN.de
```

Both the use of language files and CSS files is optional. The files `css/main.css` and `lang/main.<lang>` (where `<lang>` refers to the chosen language) are always included.

Additional parameters can be passed to the function `show_html_header`. A call with all possible parameters might look like this:

```
show_html_header "Title" "refresh=$time;url=$url;cssfile=$cssfile;showmenu=no"
```

Any additional parameters must, as shown in the example, be enclosed with quotation marks and separated by a semicolon. The syntax will *not* be checked! So it is necessary to pay close attention to the exact parameter syntax.

Here is a brief overview of the function of the parameters:

- `refresh=`*time*: Time in seconds in which the page should be reloaded by the browser.
- `url=`*url*: The URL which is reloaded on a refresh.
- `cssfile=`*cssfile*: Name of a CSS file if it differs from the name of the CGI.
- `showmenu=no`: By using this the display of the menu and the header can be suppressed.

Other Content Guidelines:

- Don't write novels, use short desriptions :-)
- Use clean HTML (SelfHTML[8] is a good starting point)
- Omit the bells and whistles (JavaScript is OK, if it does not interfere and support the user, everything also has to work without JavaScript)

The Function `show_html_footer`

The function `show_html_footer` closes the block from the CGI script which was openend by the function `show_html_header`.

[8] see http://de.selfhtml.org/

The Function `show_tab_header`

For good looking content of your generated webpage generated by the CGI you may use the `cgi-helper` function `show_tab_header`. It creates clickable "Tabs" in order to present your page divided into multiple logically separated areas.

Parameters are always passed in pairs to the show_tab_header function. The first value reflects the title of a tab, the second reflects the link. If the string "no" is passed as a link only the title will be created and the tab is not clickable (and blue).

In the following example a "window" with the title "A great window" is generated. In the window is "foo bar":

```
show_tab_header "A great window" "no"
echo "foo"
echo "bar"
show_tab_footer
```

In this example, two clickable selection tabs are generated that pass the variable `action` to the script, each with a different value.

```
show_tab_header "1st selection tab" "$myname?action=dothis" \
                "2nd selection tab" "$myname?action=dothat"
echo "foo"
echo "bar"
show_tab_footer
```

Now the script can change the content of the variable `FORM_action` (see variable evaluation below) and provide different content depending on the selection. For the clicked tab to appear selected and not clickable anymore, a "no" would have to be passed to the function instead of the link. But there is an easier way, if you hold to the convention in the following example:

```
_opt_dothis="1st selection tab"
_opt_dothat="2nd selection tab"
show_tab_header "$_opt_dothis" "$myname?action=opt_dothis" \
                "$_opt_dothat" "$myname?action=opt_dothat"
case $FORM_action in
    opt_dothis) echo "foo" ;;
    opt_dothat) echo "bar" ;;
esac
show_tab_footer
```

Hence, if a variable whose name equals the content of the variable `action` with a leading underscore (_) is passed as the title then the tab will be displayed selected.

The Function `show_tab_footer`

The function `show_tab_footer` closes the block in the CGI script that was opened by the function `show_tab_header`.

Multi-Language Capabilities

The helper script `cgi-helper` furthermore contains functions to create multi-langual CGI scripts. You only have to use variables with a leading underscore (_) for all text output. This variables have to be defined in the respective language files.

Example:

Let `lang/opt.de` contain:

```
_opt_dothis="Eine Ausgabe"
```

Let `lang/opt.en` contain:

```
_opt_dothis="An Output"
```

Let `admin/opt.cgi` contain:

```
...
echo $_opt_dothis
...
```

Form Evaluation

To process forms you have to know a few more things. Regardless of using the form's `GET` or `POST` methods, after including the `cgi-helper` script (which in turn calls the utility proccgi) the parameters can be accessed by variables named `FORM_<Parameter>`. If i.e. the form field had the name "input" the CGI script can access its content in the shell variable `$FORM_input`.

Further informations on the program `proccgi` can be found under http://www.fpx.de/fp/Software/ProcCGI.html.

User access rights: The Function `check_rights`

At the beginning of a CGI scripts the `check_rights` function has to be called in order to check whether a user has sufficient rights to use the script. Do this like here:

```
check_rights <Section> <Action>
```

The CGI script will only be executed if the user, who is logged in at the moment

- has all rights (`HTTPD_RIGHTS_x='all'`), or
- has all rights for the current area (`HTTPD_RIGHTS_x='<Bereich>:all'`), or
- has the right to execute the function in the current area (`HTTPD_RIGHTS_x='<Bereich>:<Aktion>'`).

The Function `show_error`

This funtion displays an error message in a red box. It expects two parameters: a title and a message. Example:

```
show_error "Error: No key" "No key was specified!"
```

The Function `show_warn`

This funtion displays a warning message in a yellow box. It expects two parameters: a title and a message. Example:

```
show_info "Warning" "No connection at the moment!"
```

The Function `show_info`

This funtion displays an information or success message in a green box. It expects two parameters: a title and a message. Example:

```
show_info "Info" "Action successfully executed!"
```

The Helper Script `cgi-helper-ip4`

Right after `cgi-helper` the helper script `cgi-helper-ip4` may be included by writing the following line:

```
. /srv/www/include/cgi-helper-ip4
```

A space between the dot and the slash is important!
The script provides helper funtions for checking IPv4 addresses.

The Function `ip4_isvalidaddr`

This function checks if a valid IPv4 address was passed. Example:

```
if ip4_isvalidaddr ${FORM_inputip}
then
    ...
fi
```

The Function `ipv4_normalize`

This function removes leading zeros from the passed IPv4 address. Example:

```
ip4_normalize ${FORM_inputip}
IP=$res
if [ -n "$IP" ]
then
    ...
fi
```

The Function `ipv4_isindhcprange`

This function checks whether the passed IPv4 address is ranged between the passed start and end addresses. Example:

```
if ip4_isindhcprange $FORM_inputip $ip_start $ip_end
then
    ...
fi
```

8.6.5. Miscellaneous

This and that (yes, also important!):

- `mini_httpd` does not protect subdirectories with a password. Each directory must contain a `.htaccess` file or a link to another `.htaccess` file.
- KISS - Keep it simple, stupid!
- This information may change at any time without prior notice!

8.6.6. Debugging

To ease debugging of a CGI script you may activate the debugging mode by sourcing the `cgi-helper` script. Set the variable `set_debug` to "yes" in order to do so. This will create a file `debug.log` which may be loaded down with the URL `http://<fli4l-Host>/admin/debug.log`. It contains all calls of the CGI script. The variable `set_debug` is not a global one, it has to be set anew for each CGI in question. Example:

```
set_debug="yes"
. /srv/www/include/cgi-helper
```

Furthermore, cURL[9] is ideal for troubleshooting, especially if the HTTP headers are not assembled correctly or the browser displays only blank pages. Also, the caching behavior of modern Web browser is obstructive when troubleshooting.

Example: Get a dump of the HTTP-Header with ("'*d*ump"', `-D`) and the normal output of the CGI `admin/my.cgi`. The "*u*ser" (`-u`) name here shall be "admin".

```
curl -D - http://fli4l/admin/my.cgi -u admin
```

8.7. Boot, Reboot, Dialin And Hangup Under fli4l

8.7.1. Boot Concept

FLI4L 2.0 should offer a clean install on a hard disk or a CompactFlash (TM) media, but also an installation on a Zip medium or the creation of a bootable CD-ROM should be possible. In addition, the hard drive version should not be fundamentally different from the one on an installation disk[10].

These requirements have been implemented by making it possible to move the files of the `opt.img` archive from the previous RAM disk to another medium, be it a partition on a hard disk or a CF medium. This second volume is mounted to `/opt` and only symbolic links are created from there to the rootfs. The resulting layout in the root file system then corresponds to the one unpacked in the `opt` directory of the fli4l distribution with one exception – the `files` prefix is not applicable. The file `opt/etc/rc` is then found directly under `/etc/rc`, `opt/files/bin/busybox` under `/bin/busybox`. It can be ignored that these files may be only links to a directory mounted read only as long as you do not want to modify them. If you want to do this, you have to make the files writable before by using `mk_writable` (see below).

[9] see http://de.wikipedia.org/wiki/CURL

[10] Originally fli4l could be operated from a single floppy disk. This is no longer supported since it became too big in file size.

8.7.2. Start And Stop Scripts

Scripts intended to be executed on system boot are located in the directories `opt/etc/boot.d/` and `opt/etc/rc.d/` and will also get executed in this sequence. Furthermore, scripts executed on shutdown are to be found in `opt/etc/rc0.d/`.

Important: *These script must not contain an "exit", because no separate process is created for their execution. This command would lead to a premature ending of the boot process!*

Start Scripts in `opt/etc/boot.d/`

Scripts located in this directory are executed at first. They mount the boot volume, parse the config file `rc.cfg` located on the boot medium and unpack the `opt` archive. Depending on the boot type (Page 24) these scripts are more or less complex and do the following things:

- Loading of hardware drivers (optional)
- Mount the `boot` volume (optional)
- Read the config file `rc.cfg` off the boot volume and write it to the file `/etc/rc.cfg`
- Mount the `opt` volume (optional)
- Unpack the `opt` archive (optional)

To make the scripts aware of the fli4l configuration, the configuration file `/etc/rc.cfg` is also integrated in the Rootfs archive. The configuration variables in this file are parsed by the start scripts in `opt/etc/boot.d/`. After mounting the `boot` volume `/etc/rc.cfg` is replaced by the configuration file there, so that the the current configuration of the `boot` volume is available for startup scripts in `opt/etc/rc.d/` (see below). [11]

Start Scripts in `opt/etc/rc.d/`

Commands that are executed at every start of the router can be stored in the directory `opt/etc/rc.d/`. The following conventions apply:

1. Name conventions:

   ```
   rc<three-digit number>.<Name of the OPT>
   ```

   The scripts are started in ascending order of the numbers. If multiple scripts have the same number assigned, they will be sorted alphabetically at that point. In case that the start of a package is dependant on another one, this is the determined by the number.

   Here's a general outline which numbers should be used for which tasks:

2. These scripts *must* contain all functions changing the RootFS, ie. creating of a directory `/var/log/lpd`.

[11] Normally, these two files are identical. Discrepancies are possible only if the configuration file on the `boot` volume was edited manually, for example to modify the configuration later on without the need to rebuild the fli4l archives.

Number	Task
000-099	Base system (hardware, time zone, file system)
100-199	Kernel modules (drivers)
200-299	External connections (PPPoE, ISDN4Linux, PPtP)
300-399	Network (Routing, Interfaces, Packet filter)
400-499	Server (DHCP, HTTPD, Proxy, a.s.o.)
500-900	Any
900-997	Anything causing a dialin
998-999	reserved (please do not use!)

3. These scripts shall *not* contain writing to files that could be part of the `opt` archive, because these files could be located on a volume mounted in read-only mode. If you have to modify such a file, you have to make it writeable before by using the function `mk_writable` (see below). This will create a writable copy of the file in the RootFS if needed. If the file is already writable the call of `mk_writable` will have no effect.

   Important: *`mk_writable` has to be applied directly to files in RootFS, not indirectly via the `opt` directory. If, for example, you want to modify `/usr/local/bin/foo`, the function `mk_writable` has to be called with the argument `/usr/local/bin/foo`.*

4. Before executing the actual commands these scripts have to check for the associated OPT really being active. This is usually done by a simple if-case:

   ```
   if [ "$OPT_<OPT-Name>" = "yes" ]
   then
       ...
       # Start OPT here!
       ...
   fi
   ```

5. For easier debugging the scripts should be enclosed in `begin_script` and `end_script`:

   ```
   if [ "$OPT_<OPT-Name>" = "yes" ]
   then
       begin_script FOO "configuring foo ..."
       ...
       end_script
   fi
   ```

   Debugging of start-scripts may be activated simply via `FOO_DO_DEBUG='yes'`.

6. All configuration variables are available to the scripts in direct. Explanations how to access configuration variables in scripts can be found in the section "Working with configuration variables" (Page 316).

7. The path `/opt` may not be used for storing OPT data. If in need of additional file space you should enable the user to define a suitable location by using a configuration variable. Depending on the type of data to be stored (persistent or transient data) different default

assignments should be used. A path under `/var/run/` makes sense for transient data, while for persistent data it is advised to use the function map2persistent (Page 316) combined with a suitable configuration variable.

Stop Scripts in `opt/etc/rc0.d/`

Each machine must be shut down or restarted from time to time. It is perfectly possible that you have to perform operations before the computer is shut down or restarted. To shut down and restart the commands "halt" or "reboot" are used. These commands are also invoked when the corresponding buttons in IMONC or the Web GUI are clicked.

All stop scripts can be found in the `opt/etc/rc0.d/`. The file names have to be created using the same rules as for the scripts. They are as well executed in *ascending* order of numbers.

8.7.3. Helper Functions

/etc/boot.d/base-helper provides a number of different functions that can be used in Start- and other scripts. They contian support for debugging, loading of kernel modules, or message output. The functions are listed and explained in short below

Script Control

`begin_script <Symbol> <Message>`: Output of a message and activation of script debugging by calling `set -x`, if `<Symbol>_DO_DEBUG` is set to "yes".

`end_script`: Output of an end-message and deactivation of debugging if it was activated with `begin_script`. For each `begin_script` call a corresponding `end_script` call has to exist (and vice versa).

Loading Of Kernel Modules

`do_modprobe [-q] <Modul> <Parameter>*`: Loads a kernel module including its parameters (if needed) while resolving its module dependencies. The parameter "-q" prevents error messages to be written to the console and to the boot log in case of failure. The function returns 0 for success and another value in case of error. This enables you to create code for handling failures while loading kernel modules:

```
if do_modprobe -q acpi-cpufreq
then
    # no CPU frequency scaling via ACPI
    log_error "CPU frequency scaling via ACPI not available!"
    # [...]
else
    log_info "CPU frequency scaling via ACPI activated."
    # [...]
fi
```

`do_modrobe_if_exists [-q] <Module path> <Module> <Parameter>*`:
Checks if the module `/lib/modules/<Kernel-Version>/<Module path>/<Module>` exists and, if so, invokes `do_modprobe`.

Important: *The module has to exist exactly by this name, no aliases may be used. When using an alias* `do_modprobe` *will be called immediately.*

Messages And Error Handling

`log_info <Message>`: Logs a message to the console and to `/bootmsg.txt`. If no message is passed as a parameter `log_info` reads the default input. The function always returns 0.

`log_warn <Message>`: Logs a warning message to the console and to `/bootmsg.txt`, using the string `WARN:` as a prefix. If no message is passed as a parameter `log_warn` reads the default input. The function always returns 0.

`log_error <Message>`: Logs an error message to the console and to `/bootmsg.txt`, using the string `ERR:` as a prefix. If no message is passed as a parameter `log_warn` reads the default input. The function always returns a non-zero value.

`set_error <Message>`: Output of an error message and setting of an internal error variable which can be checked later via `is_error`.

`is_error`: Clears the internal error variable and returns true if it was set before via `set_error`.

Network Functions

`translate_ip_net <Value> <Variable name> [<Result variable>]`:
Replaces symbolic references in parameters. At the moment the following translations are supported:

`*.*.*.*, none, default, pppoe` will not be translated

`any` will be replaced by `0.0.0.0/0`

`dynamic` will be replaced by the IP address of the router which represents the Internet connection

`IP_NET_x` will be replaced by the network found in the configuration

`IP_NET_x_IPADDR` will be replaced by the IP address found in the configuration

`IP_ROUTE_x` will be replaced by the routed network found in the configuration

`@<Hostname>` will be replaced by the Hosts IP address specified in the configuration

The result of the translation is stored in the variable whose name is passed in the third parameter, if this parameter is missing, the result is stored in the variable `res`. The variable name that is passed in the second parameter is used only for error messages if the translation fails, to enable the caller to pass the source of the value to be translated. In case of failure a message like

```
Unable to translate value '<Value>' contained in <Variable name>.
```

will be printed.

The return value is 0 in case of success, and unequal to zero in case of errors.

Miscellaneous

`mk_writable <File>`: Ensures that the given file is writable. If the file is located on a volume mounted in read-only mode and is only linked to the file system via a symbolic link, a local copy will be created which is then written to.

`unique <List>`: Removes duplicates from a list passed. The result is returned in the variable `list`.

8.7.4. ttyI Devices

For ttyI devices (`/dev/ttyI0` ... `/dev/ttyI15`) used by the "modem emulation" of the ISDN card a counter exists to avoid conflicts between multiple packages using these devices. For this purpose the file `/var/run/next_ttyI` is created on router start which can be queried and incremented by the OPTs. The following example script can query this value, increment it by one and export it again for the next OPT.

```
ttydev_error=
ttydev=$(cat /var/run/next_ttyI)
if [ $ttydev -le 16 ]
then                                    # ttyI device available? yes
    ttydev=$((ttydev + 1))              # ttyI device + 1
    echo $ttydev >/var/run/next_ttyI    # save it
else                                    # ttyI device available? no
    log_error "No ttyI device for <Name of your OPT> available!"
    ttydev_error=true                   # set error for later use
fi

if [ -z "$ttydev_error" ]               # start OPT only if next tty device
then                                    # was available to minimize error
    ...                                 # messages and minimize the
                                        # risk of uncomplete boot
fi
```

8.7.5. Dialin And Hangup Scripts

General

After dialin resp. hangup of a dial-up connection the scripts placed in `/etc/ppp/` are executed. OPTs may store actions here that have to be executed after connecting resp. hanging up of a connection. The name scheme for the files is as follows:

```
ip-up<three-digit number>.<OPT-Name>
ip-down<three-digit number>.<OPT-Name>
```

`ip-up` scripts will be excuted after *establishing* and `ip-down` scripts after *hangig up* of the connection.

Important: *In `ip-down` scripts no actions may be taken that lead to another dialin because this would create a permanent-online condition not desired for users without a flatrate.*

Important: *Since no separate process is created for these scripts, they may* not *invoke "exit" as well!*

Hint: If a script wants to check for `ip-up` scripts being executed the variable `ip_up_events` may be sourced from `rc400` and up. If it is set to "yes" dialup-connections exist and `ip-up` scripts will be executed. No dialup-connections are configured if it is set to "no" and `ip-up` scripts will not get executed. There is an exeception to this rule: If an Ethernet router is configured without dialup-connections but a default-Route (`0.0.0.0/0`) exists, `ip-up` scripts will get executed only once at the end of the boot process. (And as well the `ip-down` scripts on rooter shutdown.)

Variables

Due to the special call concept of the `ip-up` and `ip-down` scripts the following variables apply:

`real_interface`	the real interface, ie. `ppp0`, `ippp0`, ...
`interface`	the IMOND interface, ie. `pppoe`, `ippp0`, ...
`tty`	terminal connected, may be empty!
`speed`	connection speed, for ISDN ie. 64000
`local`	own IP address
`remote`	IP address of the Point-To-Point partner
`is_default_route`	specifies if the current `ip-up`/`ip-down` is for the interface of the default route (may be "yes" or "no")

Default Route

As of version 2.1.0 `ip-up`/`ip-down` scripts are executed for all connections, not only for the interface of the default route. To emulate the old behaviour you have to include the following in `ip-up` and `ip-down` scripts:

```
# is a default-route-interface going up?
if [ "$is_default_route" = "yes" ]
then
    # actions to be taken
fi
```

Of course, the new behaviour can also be used for specific actions.

8.8. Package "template"'

To illustrate some of the things described before the fli4l distribution provides the package "template"'. This explains by small examples how:

- a configuration files has to look like (`config/template.txt`)
- a check files is designed (`check/template.txt`)
- the extended checking mechanisms are used (`check/template.ext`)

- configuration variables are stored for later use (`opt/etc/rc.d/rc999.template`)
- stored configuration variables are processed (`opt/files/usr/bin/template_show_config`)

8.9. Structure of the Boot Medium

As of version 1.5 the program `syslinux` is used for booting. Its advantage is that a DOS-compatible file system is available on the boot medium.

The boot medium contains the following files:

`ldlinux.sys`	the "boot loader"' `syslinux`
`syslinux.cfg`	config file for `syslinux`
`kernel`	Linux kernel
`rootfs.img`	RootFS: programs needed for booting
`opt.img`	Optional files: drivers and packages
`rc.cfg`	config file containing the variables from all files in fli4l's configuration directory
`boot.msg`	Text for the `syslinux` boot menu
`boot_s.msg`	Text for the `syslinux` boot menu
`boot_z.msg`	Text for the `syslinux` boot menu
`hd.cfg`	config file to assign partitions

The script `mkfli4l.sh` (resp. `mkfli4l.bat`) at first generates the files `opt.img`, `syslinux.cfg` and `rc.cfg` as well as `rootfs.img`. The files needed are determined by the program `mkfli4l` (in the `unix`- resp. `windows`-directory). The kernel and other packages are included in those archives. The file `rc.cfg` can be found as well in the Opt-archive as on the boot medium.[12]

Subsequently, the files `kernel`, `rootfs.img`, `opt.img` and `rc.cfg` together with the `syslinux`-files are copied to the disk.

During boot fli4l uses the script `/etc/rc` to evaluate the file `rc.cfg` and integrate the compressed `opt.img`-archive into the RootFS-RAM-Disk (depending on the installation type the files are extracted directly into the rootfs ramdisk or integrated via symbolic links). Then the scripts in `/etc/rc.d/` are run in alphanumeric order and thus the drivers are loaded and all services get started.

8.10. Configuration Files

Here a short list of the files generated by fli4l "on-the-fly" at boot time.

1. Provider configuration
   - `etc/ppp/pap-secrets`
   - `etc/ppp/chap-secrets`

[12] The one in the Opt-archive is needed during early boot, because no boot volume is mounted at that time.

2. DNS configuration
   - `etc/resolv.conf`
   - `etc/dnsmasq.conf`
   - `etc/dnsmasq_dhcp.conf`
   - `etc/resolv.dnsmasq`
3. Hosts-File
   - `etc/hosts`
4. imond-configuration
   - `etc/imond.conf`

8.10.1. Provider Configuration

For the providers chosen User-ID and password are adapted in `etc/ppp/pap-secrets`.

Example for Provider Planet-Interkom:

```
# Secrets for authentication using PAP
# client        server  secret                  IP addresses
"anonymer"      *       "surfer"                *
```

In this example "anonymer"' is the USER-ID. As a remote server in principle anybody is allowed (hence "*"'). "surfer"' is the password for the Provider Planet-Interkom.

8.10.2. DNS Configuration

You can use fli4l as a DNS server. Why this is meaningful (and for Windows PCs in the LAN even mandatory) is explained in the documentation of the "base"' package.

The resolver file `etc/resolv.conf` contains the domain name and the name server to use. It has the following contents (where "domain.de"' only is a placeholder for the value of the configuration variable `DOMAIN_NAME`):

```
search domain.de
nameserver 127.0.0.1
```

The DNS server dnsmasq is configured by the file `etc/dnsmasq.conf`. It is automatically generated during boot by processing the scripts `rc001.base` and `rc370.dnsmasq` and might look like this:

```
user=dns
group=dns
resolv-file=/etc/resolv.dnsmasq
no-poll
no-negcache
bogus-priv
log-queries
domain-suffix=lan.fli4l
local=/lan.fli4l/
domain-needed
```

```
expand-hosts
filterwin2k
conf-file=/etc/dnsmasq_dhcp.conf
```

8.10.3. Hosts File

This file contains a mapping of host names to IP addresses. This assignment, however, is used only locally on the flil4 and is not visible for other computers in the LAN. This file is actually redundant if a local DNS server is started in addition.

8.10.4. imond Configuration

The file `etc/imond.conf` is constructed amongst others from the configuration variables `CIRC_x_NAME`, `CIRC_x_ROUTE`, `CIRC_x_CHARGEINT` and `CIRC_x_TIMES`. It can consist of up to 32 lines (except for comment lines). Each line has eight columns:

1. Range weekday to weekday
2. Range hour to hour
3. Device (`ippp`X or `isdn`X)
4. Circuit with default route: “yes"’/“no"’
5. Phone number
6. Name of the circuits
7. Phone charges per minute in Euros
8. Charge interval in seconds

Here an example:

```
#day  hour  device  defroute  phone        name        charge  ch-int
Mo-Fr 18-09 ippp0   yes       010280192306 Addcom      0.0248  60
Sa-Su 00-24 ippp0   yes       010280192306 Addcom      0.0248  60
Mo-Fr 09-18 ippp1   yes       019160       Compuserve  0.019   180
Mo-Fr 09-18 isdn2   no        0221xxxxxxx  Firma       0.08    90
Mo-Fr 18-09 isdn2   no        0221xxxxxxx  Firma       0.03    90
Sa-Su 00-24 isdn2   no        0221xxxxxxx  Firma       0.03    90
```

Further explanations for Least-Cost-Routing can be found in the documentation of the package “base"’.

8.10.5. The File `/etc/.profile`

The file `/etc/.profile` contains user-defined settings for the shell. To overwrite the default settings you have to create a file `etc/.profile` below the configuration directory. You may enter settings for the command prompt or abbreviations (so-called “Aliases"’) here.

Important: *This file may not contain an* `exit`*!*

Examples:

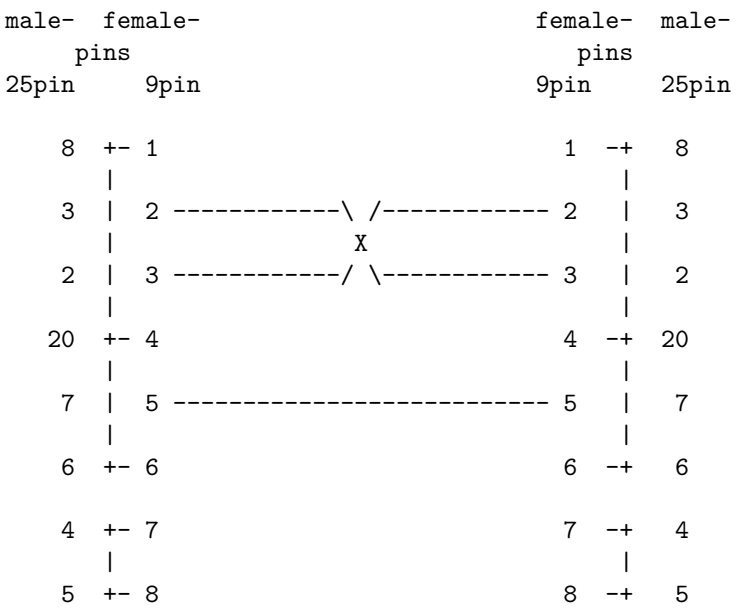
```
alias ll='ls -al'
```

A. Appendix to basepackage

A.1. Null Modem Cable

For using the otional package PPP (Page 183) a null modem cable is needed.

It needs at least three wires. This is the pin layout:



The plugs have to be soldered with the bridges shown above.

A.2. Serial Console

fli4l can be used without monitor and keyboard. A drawback of this setup is that eventual error messages will not get noticed because not all messages can be piped to the syslog-port.

A possible solution is redirecting of all console messages to a PC or a classic terminal using the serial port of the router. Configuration is done by the variables SER_CONSOLE (Page 29), SER_CONSOLE_IF (Page 29) and SER_CONSOLE_RATE (Page 29).

Machines with older mainboards/cards don not support higher serial speeds than 38400 Bd. This is why you should try with a maximum of 38400 Bd at first before testing higher port speeds. Since only text messages are displayed on the console higher speeds are not evne necessary.

All messages that usually would go to the console are now redirected to the serial port – also messages of the boot process!

As a cable to the terminal or PC with terminal emulation a Null Modem Cable (Page 338) is used. Using a standard null modem cable is discouraged because these have bridges on the handshake wires. If Terminal or PC are powered off (or no terminal emulation is loaded) the use of a standard null modem cable can thus lead to a hangup.

This is why a special wiring is needed here for using fli4l also when the terminal is deactivated. You need a 3-wire cable, with some bridges on the plug. See Nullmodemkabel (Page 338).

A.3. Programs

To save space on the boot media the program "BusyBox" is used. Ti is a single executable containing the standard Unix programs

```
[, [[, arping, ash, base64, basename, bbconfig, blkid, bunzip2, bzcat, bzip2,
cat, chgrp, chmod, chown, chroot, cmp, cp, cttyhack, cut, date, dd, df,
dirname, dmesg, dnsdomainname, echo, egrep, expr, false, fdflush, fdisk, find,
findfs, grep, gunzip, gzip, halt, hdparm, head, hostname, inetd, init, insmod,
ip, ipaddr, iplink, iproute, iprule, iptunnel, kill, killall, klogd, less, ln,
loadkmap, logger, ls, lsmod, lzcat, makedevs, md5sum, mdev, mkdir, mknod,
mkswap, modprobe, mount, mv, nameif, nice, nslookup, ping, ping6, poweroff,
ps, pscan, pwd, reboot, reset, rm, rmmod, sed, seq, sh, sleep, sort, swapoff,
swapon, sync, sysctl, syslogd, tail, tar, test, top, tr, true, tty, umount,
uname, unlzma, unxz, unzip, uptime, usleep, vi, watch, xargs, xzcat, zcat
```

. These are mostly "minimalistic" implementations which do not cover the full functional range but fully reflect the modest requirements of fli4l.

BusyBox is GPL'ed and its source can be obtained at
http://www.busybox.net/

A.4. Other i4l-Tools

There are other tools for isdn4linux that could be used for fli4l. It could be that isdnlog is more adequate as a tool to compute online-fees but it's size is 10 times higher than imond's which additionally does monitoring, controlling and Least-Cost-Routing.

A.5. Debugging

Console-Outputs are most helpful for hunting bugs. But these go by so fast on the screen, don't they? Hint: SHIFT-PAGE-UP scrolls back, SHIFT-PAGE-DOWN scroll forwards.

If the error message "try-to-free-pages" occurs during router use there is not enough RAM left for the programs. Try the following options to recover:

- add more RAM
- use less Opt-Packages
- try a harddisk-installation according to Typ B (Page 15)

proc-files can help debugging, for example executing

```
cat /proc/interrupts
```

shows the interrupts used by the drivers – not those used by the hardware!

More interesting files under /proc are devices, dma, ioports, kmsg, meminfo, modules, uptime, version and pci (if the router has a PCI-Bus).

Often a connection problem with ipppd is caused by failing authentification. The variables

```
OPT_SYSLOGD='yes'

OPT_KLOGD='yes'
```

in config/base.txt and

```
ISDN_CIRC_x_DEBUG='yes'
```

in config/isdn.txt can help here.

A.6. Literature

- Computer Networks, Andy Tanenbaum
- TCP/IP Netzanbindung von PCs, Craig Hunt
- TCP/IP, Kevin Washburn, Jim Evans, Verlag: Addison-Wesley, ISBN: 3-8273-1145-4
- TCP/IP Netzanbindung von PCs, ISBN 3-930673-28-2
- TCP/IP Netzwerk Administration, ISBN 3-89721-110-6
- Linux-Anwenderhandbuch, ISBN 3-929764-06-7
- TCP/IP im Detail: http://www.nickles.de/c/s/ip-adressen-112-1.htm
- Generell das online Linuxanwenderhandbuch von Lunetix unter: http://www.linux-ag.com/LHB/
- Einführung in die Linux-Firewall: http://www.little-idiot.de/firewall/

A.7. Prefixes

For units, prefixes addressed in this document are according to IEC 60027-2.
See: http://physics.nist.gov/cuu/Units/binary.html.

A.8. Warranty and Liability

There is no warranty and liability whatsoever for the whole fli4l distribution or parts of it. Also there is no guarantee for function or correct documentation whereever you may find it.

There is no Liability at all for eventual damages or costs that may arise! In other words: Don't complain if it eats your hamster.

A.9. Credits

In this part of the documentation all people are honored that contribute or have contributed to the development of fli4l.

A.9.1. Foundation Of The Project

Meyer, Frank email: frank(at)fli4l(dot)de

Frank started the Projekt fli4l on May, 4th 2000!
See: http://www.fli4l.de/home/eigenschaften/historie/

A.9.2. Developer- and Testteam

The fli4l-Team consists of (in alphabetical order):

Eckhofer, Felix (*Dokumentation, Howtos*)
email: felix(at)fli4l(dot)de
Franke, Roland (*OW, FBR*)
email: fli4l(at)franke-prem(dot)de
Hilbrecht, Claas (*VPN, Kernel*)
email: claas(at)jucs-kramkiste(dot)de
Klein, Sebastian (*Kernel, Wlan*)
email: -
Knipping, Michael (*Accounting*)
email: fli4l(at)knibo(dot)de
Krister, Stefan (*Opt-Cop, lcd4linux*)
email: stefan(dot)krister(at)chreativ(dot)chaos(dot)de
Miksch, Gernot (*LCD*)
email: gernot_miksch(at)gmx(dot)de
Schiefer, Peter (*fli4l-CD, Opt-Cop, Webseite, Releasemanagement*)
email: peter(at)fli4l(dot)de
Schulz, Christoph (*FBR, IPv6, Kernel*)
email: fli4l(at)kristov(dot)de
Siebmanns, Harvey (*Dokumentation*)
email: -
Spieß, Carsten (*Dsltool, Hwsupp, Rrdtool, Webgui*)
email: fli4l(at)carsten-spiess(dot)de
Vosselman, Arwin (*LZS-Kompression, Dokumentation*)
email: arwin(at)xs4all(dot)nl
Wallmeier, Nico (*Windows-Imonc*)
email: nico(at)fli4l(dot)de
Walter, Gerd (*UMTS*)
email: fli4l(at)hgwb(dot)de
Walter, Oliver (*QoS*)
email: owb(at)gmx(dot)de
Weiler, Manuela (*CD-Versand, Kassenwart*)
email: -

Weiler, Marcel (*Qualitätsmanagement*)
email: -
Wolters, Florian (*Firmware, Kernel*)
email: fli4l(at)florian-wolters(dot)de

The fli4l-Test- and Translation-team consists of (in alphabetical order):

Bußmann, Lars
Charrier, Bernard
Fischer, Joerg
Frauenhoff, Peter
Schliesing, Manfred
Schmitts, Jupp

A.9.3. Developer- and Testteam (inactive)

Arndt, Kai-Christian (*USB*)
Behrends, Arno (*Support*)
Bork, Thomas (*lpdsrv*)
Bauer, Jürgen (*LCD-Package, fliwiz*)
Blokland, Kees (*Englische Übersetzung*)
Cerny, Carsten (*Webseite, fliwiz*)
Dawid, Oliver (*dhcp, uClibc*)
Ebner, Hannes (*QoS*)
Grabner, Hans-Joerg (*imonc*)
Grammel, Matthias (*Englische Übersetzung*)
Gruetzmacher, Tobias (*Mini-httpd, imond, proxy*)
Hahn, Joerg (*IPSEC*)
Hanselmann, Michael (*Mac OS X/Darwin*)
Hoh, Jörg (*Newsletter, NIC-DB, Veranstaltungen*)
Hornung, Nicole (*Verein*)
Horsmann, Karsten (*Mini-httpd, WLAN*)
Janus, Frank (*LCD*)
Kaiser, Gerrit (*Logo*)
Karner, Christian (*PPTP-Package*)
Klein, Marcus (*Problemfeedback*)
Lammert, Gerrit (*HTML-Dokumentation*)
Lanz, Ulf (*LCD*)
Lichtenfeld, Nils (*QoS*)
Neis, Georg (*fli4l-CD, Dokumentation*)
Peiser, Steffen (*FAQ*)
Peus, Christoph (*uClibc*)
Pohlmann, Thorsten (*Mini-httpd*)
Raschel, Tom (*IPX*)
Reinard, Louis (*CompactFlash*)
Resch, Robert (*PCMCIA, WLAN*)
Schäfer, Harald (*HDD-Support*)
Strigler, Stefan (*GTK-Imonc, Opt-DB, NG*)
Wolter, Jean (*Paketfilter, uClibc*)
Zierer, Florian (*Wunschliste*)

A.9.4. Sponsors

Meanwhile fli4l is a registered als Word-/trademark. The following fli4l-Users (there are more that did not want to be mentioned) have helped to raise the money needed for this:

Bebensee, Norbert
Becker, Heiko
Behrends, Arno
Böhm, Stefan
Brederlow, Ralf
Groot, Vincent de
Hahn, Olaf
Hogrefe, Paul
Holpert, Christian
Hornung, Nicole
Kuhn, Robert
Lehnen, Jens
Ludwig, Klaus-Ruediger
Mac Nelly, Christa
Mahnke, Hans-Jürgen
Menck, Owen
Mende, Stefan
Mücke, Michael
Roessler, Ingo
Schiele, Michael
Schneider, Juergen
Schönleber, Suitbert
Sennewald, Matthias
Sternberg, Christoph
Vollmar, Thomas
Walter, Oliver
Wiebel, Christian
Woelk, Fabian

For some time fli4l has its own sponsors, whose (Hardware-)donations have helped to support the fli4l development. In detail this are adapters, CompactFlash and Ethernet cards.

Hardware-donours (in alphabetical order):

Baglatzis, Stephanos
Bauer, Jürgen
Dross, Heiko
Kappenhagen, Wenzel
Kipka, Joachim
Klopfer, Tom
Peiser, Steffen
Reichelt, Detlef
Reinard, Louis
Stärkel, Christopher

More sponsors can be found on the fli4-homepage:
http://www.fli4l.de/sonstiges/sponsoren/

A.10. Feedback

Critics, feedback and cooperation are always welcome.

The primary point of contact are the fli4l-Newsgroups. Those having problems in the setup of a fli4l-Routers, should at first read the FAQ, Howtos and the NG-Archives, before posting in the newsgroups. Informations on the different groups and netiquette can be found on the fli4l-Webseite:

http://www.fli4l.de/hilfe/newsgruppen/
http://www.fli4l.de/hilfe/faq/
http://www.fli4l.de/hilfe/howtos/

Because mostly older hardware is used for fli4l problems may be inevitable. Informations can help other fli4l-users having hardware problems with PC-Cards (I/O-Addresses, Interrupts and so on).

On fli4l's website is a link to a network card database, where appropriate drivers for certain cards are listed and can be entered.

http://www.fli4l.de/hilfe/nic-db/

Have fun with fli4l!

B. Appendixes to optional packages

B.1. CHRONY - Inform other applications about timewarps

If chrony notes that the clock is significantly away from the current time, it corrects the time in one great step and starts scripts to inform other applications about this timewarp. For example to inform imond about a timewarp, chrony does the following:

1. include scripts into the archive

   Chrony includes two files to the archive:

   ```
   start_imond  yes  etc/chrony.d/timewarp.sh mode=555 flags=sh
   start_imond  yes  etc/chrony.d/timewarp100.imond mode=555 flags=sh
   ```

   timewarp.sh starts all scripts in the same folder which names are timewarp<3 numbers>.<name>.

2. provide script

   chrony includes the following script into the archive:

   ```
   # inform imond about time warp
   imond-stat "adjust-time $timewarp 1"
   ```

   Hence imond will be informed about the timewarp and is able to correct it's internal timebase.

B.2. DSL - PPPD and Active Filter

fli4l uses the expression:

```
'outbound and not icmp[0] != 8 and not tcp[13] & 4 != 0'
```

and accomplishes that generally only packets sent from the local network to the internet keep the connection open, with a few exceptions:

- *TCP-RST*: Answers to rejected connection from outside do not reset the timeout,
- *ICMP*: ICMP messages sent do not reset the timeout unless an echo request is sent.

This expression is converted by the PPPD into a packet-filter usable by the kernel. In this example it looks like this:

```
#
# Expression: outbound and not icmp[0] != 8 and not tcp[13] & 4 != 0
#
(000) ldb      [0]
(001) jeq      #0x0             jt 17   jf 2
(002) ldh      [2]
(003) jeq      #0x21            jt 4    jf 18
(004) ldb      [13]
(005) jeq      #0x1             jt 6    jf 11
(006) ldh      [10]
(007) jset     #0x1fff          jt 18   jf 8
(008) ldxb     4*([4]&0xf)
(009) ldb      [x + 4]
(010) jeq      #0x8             jt 18   jf 17
(011) jeq      #0x6             jt 12   jf 18
(012) ldh      [10]
(013) jset     #0x1fff          jt 18   jf 14
(014) ldxb     4*([4]&0xf)
(015) ldb      [x + 17]
(016) jset     #0x4             jt 17   jf 18
(017) ret      #0
(018) ret      #4
```

B.3. DYNDNS

B.3.1. Adding Of New Providers

Adding new providers is easy because update-scripts are separated from provider data completely. For a new provider adapt the following files:

opt/etc/dyndns/provider.NAME

This file defines how an update is working with this provider. It mostly consists only of a list of variables but is a normal shell script that even allows complex operations to be done. This should not be necessary in most cases. These variables can be used in the file:

$ip The IP of the interface that should get the dynamic hostname.

$host The complete hostname the user specified in his configuration.

$subdom All components of the hostname ending with the dot next to last (**name**.provider.dom)

$domain the both last components of the hostname (name.**provider.dom**)

$provider The symbolic name of the provider the user specified in his configuration file.

$user The username for this service.

$pass The password.

These variables can be put in curly brackets to be cleary distinguishable from normal text, `$ip` i.e. becomes `${ip}`. If using quotation marks it should be noted that within single quotes the variables mentioned above are *not* expanded while this works with double quotes. As a rule of thumb: Always use single quotes but when using variables double quotes are needed.

The following variables must be defined in this file in order to get an update working with the provider:

provider_update_type This determines the type of query sent to the provider's server. These types are supported at the moment:

- **http** A predefined website of the provider will be loaded to update the DynDNS-entry.
- **netcat** A predefined text will be sent to the provider's server triggering an update.
- **gnudip** An update process relatively easy and secure done by two HTTP-queries.

provider_host The provider's hostname that is to be contacted during an update.

provider_port The port to be contacted on the provider's host. Standard-port for HTTP is 80.

Depending on the update type further variables have to be specified:

HTTP provider_url The relative URL (without the hostname, but with an / at the beginning) for the provider script. For examples please have a look at the files for other providers.

provider_auth (optional) If the provider needs a login via basic authentication provide the information needed here. The format is `"USER:PASSWORD"`.

Netcat provider_data The text to be sent to the provider's server. See `provider.DYNEISFAIR` as an example.

GNUDip provider_script The path tot he GNUDip-script on the server, mostly something like `'/cgi-bin/gdipupdt.cgi'`.

opt/dyndns.txt

One or more lines for the new provider have to be inserted here. Usually a line like that is enough:

```
dyndns_%_provider   NAME   etc/dyndns/provider.NAME
```

If HTTP and Basic Authentication are used by the provider you will need the base64 executable:

```
dyndns_%_provider   NAME   files/usr/local/bin/base64
```

If other tools are needed sent an email so we can validate if this is suitable for OPT_DYNDNS.

check/dyndns.exp

In this file the provider name has to be added at the end of the long line starting with `DYNPROVIDER =` , seperated by a '|'.

doc/<Language>/tex/dyndns/dyndns_main.tex

Add a new paragraph to the documentation. The providers have to be sorted alphabetically by the short name given by the user in the config file. The prov-macro is documented at the beginning of the file, enough examples should be present.

B.3.2. Note Of Thanks

At first I wish to thank Thomas Müller (email: opt_dyndns@s2h.cx) who originally developed this package and maintained it for a long time. He has done exceptional work here, without him this packages would not have been possible.

I would like to thank as well Marcel Döring (email: m@rcel.to), who maintained the package for quite some time.

A lot of people have been helping and providing ideas at the development of the package. Many thanks also to all those hard-working people.

Further thanks got to all the people contributing to the package by providing tips, new providers, bug reports and so on:

Last but not least my thanks go to Frank Meyer and the rest of the fli4l team for their tireless work to tinker with the best router in the world (;-) Please do not take this too serious).

B.3.3. Licence

B.4. EASYCRON - Adding To Crontab While Booting

Important: *The following statements are intended only for developers of opt-packages for fli4l routers.*

As of version 2.1.7 Easycron's rc-script provides the function `add_crontab_entry()`. Other rc-scripts starting from rc101 to rc949 can add entries to crontab by calling this function. These additional entries get activated with the start of the cron daemon at the end of booting.

```
add_crontab_entry time command [custom]
```

Use `time` to set the execution time in cron notation (see manpage crontab(5) at http://linux.die.net/man/5/crontab). `command` contains the command to be executed. The third

parameter `custom` is optional. By using it you can set environment variables needed for the command used. If more than one variable has to be set separate them by a \\.
Please do not change the environment variable `PATH` because otherwise the following crontab entries could not be executed correctly anymore.

```
#
# example I: normal use, 2 parameters
#
     crontime="0 5 1 * *"
     croncmd="rotate_i_log.sh"

     add_crontab_entry "$crontime" "$croncmd"

#
# end of example I
#

#
# example II: extended use, 3 parameters and
#                           multiple environment values
#
     croncustom="source=/var/log/current \\ dest=/mnt/data/log"
     croncmd='cp $source $dest-`date +\%Y\%m\%d`; > $source'
     crontime="59 23 * * *"

     add_crontab_entry "$crontime" "$croncmd" "$croncustom"

#
# end of example II
#
```

The accuracy of the entries must be ensured by the calling script.

B.5. HD - Possible Errors Concerning Hardisks/CompactFlashs

Problem:

- the router does not recognize the harddisk at all.

Possible Causes:

- the router lacks drivers for the hd-controller - additional drivers for the controller may be needed. Configure `OPT_HDDRV` in this case.
- BIOS entry for the disk is wrong.
- Controller is defective or switched off.

- wrong disk is configured for the installation
- Controller is not supported by fli4l. Some controllers may need special drivers not included with fli4l

Problem:

- Installation routine stops
- after a remote update of the opt-archive the router refuses to boot
- Error messages occur while partitioning or formatting the harddisk

Possible Causes:

- IDE harddisks can suffer from cables too long or unmatching
- older harddisks may suffer from wrong PIO mode or transfer rate settings in Bios (or controller) eventually being too fast for the disk.
- IDE-Chipset not suitable

Remarks:

- Problems with DMA eventually can be solved by setting `LIBATA_NODMA='no'`. (The default is 'yes'). This activates DMA with ATA devices.

Problem:

- fli4l doesn't boot from harddisk after the installation

Possible Cause:

- If booting from a CF module fails check if the CF was recognized as LBA or LARGE by the Bios. Correct setting for modules smaller than 512MB is NORMAL or CHS.
- an Adaptec 2940 Controller with an old BIOS is used and extended mapping for harddisks over 1GB is active. Update the Bios of the SCSI controller or switch mapping.
  By switching the mapping all data on the disk will be lost!

Problem:

- Windows error message while preparing of a CF-card: „Medium in drive (X:) contains no FAT. [Cancel]".

Possible Cause:

- The card was removed from the reader too early / without unmounting. Windows did not finish writing and the file system is damaged. Prepare the CompactFlash again at the fli4l via HD-install.

B.6. HTTPD

B.6.1. Additional Settings

These variables are not present in the configuration and thus have to be added to it when needed.

HTTPD_USER By using this option the web server can be run with the rights of another user than „root". This is useful especially if the webserver should display other pages than the admin interface. **Scripts that need access to configuration files possibly won't run as expected then. Standard scripts will run with any user provided.**

B.6.2. Remarks

If TELMOND is installed the telephone numbers of callers will be displayed on the pages 'status' and 'calls'. Names can be assigned via entries in the file opt/etc/phonebook. This file has the same format as IMONC's telephone number file. Phonebooks may be exchanged between IMONC and the router. The format of each row is "Phone number=Name[,WAV-file]" WAV files will only be used by IMONC and are ignored by the web server.

The complete web interface has been converted to frameless design using css as of version 2.1.12. Really old browsers may have problems displaying this. The advantage is that the look of the pages can be changed almost completely simply by editing the css files (mainly /opt/srv/www/css/main.css).

The webserver package was devloped by Thorsten Pohlmann (email: pohlmann@tetronik.com) and is maintained by Tobias Gruetzmacher (email: fli4l@portfolio16.de) at the moment. The new design (as of version 2.1.12) was realized by Helmut Hummel (email: hh@fli4l.de).

B.7. HWSUPP - Device dependant settings

B.7.1. Available LED devices

Depending on the `HWSUPP_TYPE` different LED devices are present. For hardware not listed here the PC keyboard LEDs are available using generic-pc.

Additional LED devices can be mounted on eg. WLAN adapters. The valid LED device names can be determined by executing `ls /sys/class/leds/`, eg. via ssh on the router's console.

sim

LED simulation, will log to `syslog`:

- `simu::1`
- ...
- `simu::8`

generic-pc

PC keyboard LEDs:

- `keyboard::scroll`
- `keyboard::caps`
- `keyboard::num`

generic-acpi

PC keyboard LEDs, like generic-pc

pcengines-alix

- `alix::1`
- `alix::2`
- `alix::3`

pcengines-apu

- `apu::1`
- `apu::2`
- `apu::3`

pcengines-wrap

- `wrap::1`
- `wrap::2`
- `wrap::3`

soekris-net4801

- `net48xx::error`

soekris-net5501

- `net5501::error`

B.7.2. Available Button Devices

Depending on the `HWSUPP_TYPE` different GPIO devices are predefined for buttons.

pcengines-alix

- `gpio::24`

pcengines-apu

- `gpio::252`

pcengines-wrap

- `gpio::40`

soekris-net5501

- `gpio::25`
  The button is named 'Reset' on the soekris case.
  Attention: the button must be enabled in BIOS.

B.7.3. Hardware specific notes

pcengines-alix

A faulty driver for the lm90 temperatur sensor causes a loss of temperature monitoring.

As a workaraound the lm90 driver will be unloaded and reloaded again automatically by a cron job. This requires the package easycron to be loaded (`OPT_EASYCRON='yes'`).

B.8. HWSUPP - Configuration examples

B.8.1. generic-pc

```
OPT_HWSUPP='yes'
HWSUPP_TYPE='generic-pc'

HWSUPP_WATCHDOG='no'
HWSUPP_CPUFREQ='no'

HWSUPP_LED_N='3'
HWSUPP_LED_1='ready'
HWSUPP_LED_1_DEVICE='keyboard::num'
HWSUPP_LED_2='online'
HWSUPP_LED_2_DEVICE='keyboard::caps'
HWSUPP_LED_3='wlan'
HWSUPP_LED_3_DEVICE='keyboard::scroll'
HWSUPP_LED_3_WLAN='wlan0'

HWSUPP_BUTTON_N='0'
```

B.8.2. pcengines-apu

```
OPT_HWSUPP='yes'
HWSUPP_TYPE='pcengines-apu'
```

```
HWSUPP_WATCHDOG='yes'
HWSUPP_CPUFREQ='yes'
HWSUPP_CPUFREQ_GOVERNOR='ondemand'

HWSUPP_LED_N='3'
HWSUPP_LED_1='ready'
HWSUPP_LED_1_DEVICE='apu::1'
HWSUPP_LED_2='wlan'
HWSUPP_LED_2_DEVICE='apu::2'
HWSUPP_LED_2_WLAN='wlan0'
HWSUPP_LED_3='online'
HWSUPP_LED_3_DEVICE='apu::3'

HWSUPP_BUTTON_N='1'
HWSUPP_BUTTON_1='wlan'
HWSUPP_BUTTON_1_DEVICE='gpio::252'
HWSUPP_BUTTON_1_PARAM='wlan0'
```

B.8.3. pcengines-apu with GPIO's

```
OPT_HWSUPP='yes'
HWSUPP_TYPE='pcengines-apu'

HWSUPP_WATCHDOG='yes'
HWSUPP_CPUFREQ='yes'
HWSUPP_CPUFREQ_GOVERNOR='ondemand'

HWSUPP_LED_N='5'
HWSUPP_LED_1='ready'
HWSUPP_LED_1_DEVICE='apu::1'
HWSUPP_LED_2='wlan'
HWSUPP_LED_2_DEVICE='apu::2'
HWSUPP_LED_2_WLAN='wlan0'
HWSUPP_LED_3='online'
HWSUPP_LED_3_DEVICE='apu::3'
HWSUPP_LED_4='trigger'
HWSUPP_LED_4_PARAM='phy0rx'
HWSUPP_LED_4_DEVICE='gpio::237'
HWSUPP_LED_5='trigger'
HWSUPP_LED_5_PARAM='phy0tx'
HWSUPP_LED_5_DEVICE='gpio::245'

HWSUPP_BUTTON_N='2'
HWSUPP_BUTTON_1='wlan'
HWSUPP_BUTTON_1_DEVICE='gpio::252'
HWSUPP_BUTTON_1_PARAM='wlan0'
HWSUPP_BUTTON_2='online'
```

```
HWSUPP_BUTTON_2_DEVICE='gpio::236'
```

B.9. HWSUPP - Blink Sequences

The following blink sequences are displayed during boot:

1.	⊗				⊗				...
2.	⊗	⊗			⊗	⊗			...
3.	⊗	⊗	⊗		⊗	⊗	⊗		...
4.	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	...

The first sequence is displayed while processing `rc002.*` to `rc250.*`
(1 * blink - pause),
for `rc250.*` to `rc500.*` the second (2 * blink - pause),
for `rc500.*` to `rc750.*` the third and
for `rc750.*` until the end of the boot process the forth sequence (coninuous blinking).

B.10. HWSUPP - Hints for package developers

This chapter describes the things a package developer has to do to add LED or button functionality to a package[1].

B.10.1. LED extensions

LED type

Within the file `check/myopt.exp` the list of LED types which can be entered in `HWSUPP_LED_x` is extended.

Example:

```
+HWSUPP_LED_TYPE(OPT_MYOPT) = 'myopt'
                            : ', myopt'
```

Parameter check

Within `check/myopt.ext` the parameters which can be entered in `HWSUPP_LED_x_PARAM` will be checked.

Example:

```
if (opt_hwsupp)
then
    depends on hwsupp version 4.0

    foreach i in hwsupp_led_n
    do
        set action=hwsupp_led_%[i]
        set param=hwsupp_led_%_param[i]
```

[1] Search for `##HWSUPP##` in the WLAN package to find the places to adapt.

```
        if (action == "myopt")
        then
            if (!(param =~ "(RE:MYOPT_LED_PARAM)"))
            then
                error "When HWSUPP_LED_\${i}='myopt', ...
                       must be entered in HWSUPP_LED_\${i}_PARAM"
            fi
        fi
    done
fi
```

LED Display

The command `/usr/bin/hwsupp_setled <LED> <status>/` has to be executed to set a LED in a package script (eg. `/usr/bin/<opt>_setled`)

The LED number can be found in `/var/run/hwsupp.conf`.

Status can be `off`, `on` or `blink`.

Example:

```
if [ -f /var/run/hwsupp.conf ]
then
    . /var/run/hwsupp.conf
    [ 0$hwsupp_led_n -eq 0 ] || for i in `seq 1 $hwsupp_led_n`
    do
        eval action=\$hwsupp_led_${i}
        eval param=\$hwsupp_led_${i}_param
        if [ "$action" = "<opt>" ]
        then
            if [ <myexpression> ]
            then
                /usr/bin/hwsupp_setled $i on
            else
                /usr/bin/hwsupp_setled $i off
            fi
        fi
    done
fi
```

The actual state of a LED can be queried with `/usr/bin/hwsupp_getled <LED>/`. The result will be `off`, `on` or `blink`.

B.10.2. Button extensions

B.10.3. Button action

In `check/myopt.exp` the list of button types allowed in `HWSUPP_LED_x` can be extended.

Beispiel:

```
+HWSUPP_BUTTON_TYPE(OPT_MYOPT) = 'myopt'
                               : ', myopt'
```

Parameter check

The parameters which can be entered in `HWSUPP_BUTTON_x_PARAM` will be checked using `check/myopt.ext` .

Example:

```
if (opt_hwsupp)
then
    depends on hwsupp version 4.0

    foreach i in hwsupp_button_n
    do
        set action=hwsupp_buttonn_%[i]
        set param=hwsupp_button_%_param[i]
        if (action == "myopt")
        then
            add_to_opt "files/usr/bin/myopt_keyprog" "mode=555 flags=sh"
            if (!(param =~ "(RE:MYOPT_BUTTON_PARAM)"))
            then
                error "When HWSUPP_BUTTON_\${i}='myopt', ...
                       must be entered in  HWSUPP_BUTTON_\${i}_PARAM"
            fi
        fi
    done
fi
```

Button function

When a button is pressed the script file `/usr/bin/myopt_keyprog` will be executed.

The content of `HWSUPP_BUTTON_x_PARAM` is passed as a parameter.

Example:

```
##TODO## example
```

B.11. 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/).

B.11.1. 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.

B.11.2. 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:

```
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.

B.11.3. 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. [2]

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:

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")[3]

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.

[2] see RFC 1918(`http://tools.ietf.org/html/rfc1918`) for details
[3] in case that the router gets its IPv4-address dynamically this will specifiy "heartbeat"

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",[4] 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
```

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-prefixaund 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')
```

[4] Of course you can have another sub-subnet!

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'
```

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 “traceroute” (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.
```

B.12. ISDN

B.12.1. Technical Details About Dial-In And Routing With ISDN

This chapter is interesting for those who want to know what is happening 'under the hood', having special wishes for configuration or simply looking for solutions to their problems. All others are *not* encouraged to read this chapter.

After establishing a connection to the provider the ipppd daemon that has made the connection newly configures the interface to set the negotiated IP addresses. The linux kernel automatically sets corresponding routes for remote IP address and netmask. Existing special routes will be deleted. If no netmask is given the ipppd derives it from the remote IP netmask (Class A, B and C subnets will be used here). The vanishing of existing and appearing of new routes has raised problems time and time again:

- Company networks were unaccessible because of routes vanishing or being overlayed by newly set ones
- Interfaces were dialing in apparently without cause because a packet was routed by the kernel to another interface instead of the default route
- ...

Because of that it is tried to avoid unwanted routes.
The following things will be changed to achieve this:

- remote IP will be set to 0.0.0.0 if nothing else is specified. Hence the routes configured by the kernel while initializing the interface will vanish.
- additionally set routes will be saved in a file
- if a netmask is given for the circuit it will be transferred to the ipppd in order to use it for the configuration of the interface (and therefore route generation) after negotiation of an IP.
- after dial-in the saved routes of the circuit will be reloaded from the file and set again (they were deleted by the kernel while ipppd was reconfiguring the interface)
- after hangup the interface will be reconfigured and routes are set anew to restore the initial situation.

Configuration of a circuits looks like this in that case:

- item default route

  ```
  ISDN_CIRC_%_ROUTE='0.0.0.0'
  ```

  If the circuit is a lcr circuit and "active" in the moment a default route will be set towards the circuit (res. the according interface). After dial-in a host-route to the provider appears that vanishes after hanging up.

- special routes

  ```
  ISDN_CIRC_%_ROUTE='network/netmaskbits'
  ```

  The given routes for the circuit (res. the according interface) will be set. After dial-in the routes deleted by the kernel will be restored and a host-route to the dial-in node exists. After hangup the initial state will be restored.

- remote ip

  ```
  ISDN_CIRC_%_REMOTE='ip address/netmaskbits'
  ISDN_CIRC_%_ROUTE='network/netmaskbits'
  ```

  While configurating the interface routes to the target net appear (according to IP address AND netmask). If the specified IP is kept after dial-in (meaning no other IP is negotiated during connection establishment) the route will be kept as well.

  If another IP was negotiated during dial-in the route will change accordingly (new IP AND netmask).

  For additional routes see above.

This will hopefully solve *all* problems raised by special routes. The way of correction may change in the future but the principle won't.

B.12.2. Error Messages Of The ISDN-Subsystem (i4l-Documentation)

Following is an excerpt from the Isdn4Linux Documentation (man 7 isdn_cause).

Cause messages are 2-byte information elements, describing the state transitions of an ISDN line. Each cause message describes its origination (location) in one byte, while the cause code is described in the other byte. Internally, when EDSS1 is used, the first byte contains the location while the second byte contains the cause code. When using 1TR6, the first byte contains the cause code while the location is coded in the second byte. In the Linux ISDN subsystem, the cause messages visible to the user are unified to avoid confusion. All user visible cause messages are displayed as hexadecimal strings. These strings always have the location coded in the first byte, regardless if using 1TR6 or EDSS1. When using EDSS1, these strings are preceeded by the character 'E'.

LOCATION The following location codes are defined when using EDSS1:

00 Message generated by user.
01 Message generated by private network serving the local user.
02 Message generated by public network serving the local user.
03 Message generated by transit network.
04 Message generated by public network serving the remote user.
05 Message generated by private network serving the remote user.
07 Message generated by international network.
0A Message generated by network beyond inter-working point.

CAUSE The following cause codes are defined when using EDSS1:

01 Unallocated (unassigned) number.
02 No route to specified transit network.
03 No route to destination.
06 Channel unacceptable.
07 Call awarded and being delivered in an established channel.
10 Normal call clearing.
11 User busy.
12 No user responding.
13 No answer from user (user alerted).
15 Call rejected.
16 Number changed.
1A Non-selected user clearing.
1B Destination out of order.
1C Invalid number format.
1D Facility rejected.
1E Response to status enquiry.
1F Normal, unspecified.
22 No circuit or channel available.
26 Network out of order.
29 Temporary failure.
2A Switching equipment congestion.
2B Access information discarded.
2C Requested circuit or channel not available.
2F Resources unavailable, unspecified.
31 Quality of service unavailable.
32 Requested facility not subscribed.

39 Bearer capability not authorised.
3A Bearer capability not presently available.
3F Service or option not available, unspecified.
41 Bearer capability not implemented.
42 Channel type not implemented.
45 Requested facility not implemented.
46 Only restricted digital information bearer.
4F Service or option not implemented, unspecified.
51 Invalid call reference value.
52 Identified channel does not exist.
53 A suspended call exists, but this call identity does not.
54 Call identity in use.
55 No call suspended.
56 Call having the requested call identity.
58 Incompatible destination.
5B Invalid transit network selection.
5F Invalid message, unspecified.
60 Mandatory information element is missing.
61 Message type non-existent or not implemented.
62 Message not compatible with call state or message or message type non existent or not implemented.
63 Information element non-existent or not implemented.
64 Invalid information element content.
65 Message not compatible.
66 Recovery on timer expiry.
6F Protocol error, unspecified.
7F Inter working, unspecified.

B.13. UMTS

B.13.1. Supported Hardware

```
This package supports the follwing UMTS-hardware:
For proper operation other packages may be needed.
For USB-Adapters Package USB has to be activated.
OPT_USB='yes'

Hardware:               tested          additional packages

Novatel Adapters:
Merlin U530             yes             PCMCIA, TOOLS (serial)
Merlin U630             no              PCMCIA, TOOLS (serial)

MC950D                  yes             USB


OPTION Adapters:
3G Datacard             no              PCMCIA, USB
GT 3G Quad              yes             PCMCIA, USB
GT Fusion               no              PCMCIA, USB
```

```
GT MAX HSUPA GX0301     yes             PCMCIA, USB
for the four Cardbus-adapters set PCMCIA_PCIC='yenta_socket'

Icon 225 (GIO225)       yes             USB

Huawei Adapter:
E220, E230, E270        yes             USB
E510                    yes             USB
E800                    yes             USB
K3520                   yes             USB

ZTE Adapter:
MF110                   yes             USB
MF190                   yes             USB
```

B.13.2. Modem Interface Not Activated

For some OPTION UMTS Sticks it may occur that the Modem interface which is needed for pppd is not activated.

```
Example using GIO225 adapter

check via:
grep "" /sys/bus/usb/devices/*/tty/*/hsotype

Output should look like this:
/sys/bus/usb/devices/2-1:1.0/tty/ttyHS0/hsotype:Control
/sys/bus/usb/devices/2-1:1.0/tty/ttyHS1/hsotype:Application
/sys/bus/usb/devices/2-1:1.1/tty/ttyHS2/hsotype:Diagnostic

hsotype:Modem is missing here.

Check the interface configuration via this comman:
chat -e -t 1 '' "AT_OIFC?" OK >/dev/ttyHS0 </dev/ttyHS0

Output should look like this:
AT_OIFC?
_OIFC: 3,1,1,0

OK

If you see this
AT_OIFC?
_OIFC: 2,1,1,0

OK
```

```
you can activate the modem interface via the command:
chat -e -t 1 '' "AT_OIFC=3,1,1,0" OK >/dev/ttyHS0 </dev/ttyHS0

After that unplug the adapter and replug it afterwards.
Now check via:
grep "" /sys/bus/usb/devices/*/tty/*/hsotype

for a modem entry.
/sys/bus/usb/devices/2-1:1.0/tty/ttyHS0/hsotype:Control
/sys/bus/usb/devices/2-1:1.0/tty/ttyHS1/hsotype:Application
/sys/bus/usb/devices/2-1:1.1/tty/ttyHS2/hsotype:Diagnostic
/sys/bus/usb/devices/2-1:1.2/tty/ttyHS3/hsotype:Modem
```

B.14. Differences version 3.10.5 and version 3.6.2

Package ADVANCED_NETWORKING

New variables

BCRELAY_N (Page 76)
BCRELAY_x_IF_N (Page 76)
BCRELAY_x_IF_x (Page 76)
ETHTOOL_DEV_N (Page 86)
ETHTOOL_DEV_x (Page 86)
ETHTOOL_DEV_x_OPTION_N (Page 86)
ETHTOOL_DEV_x_OPTION_x_NAME (Page 86)
ETHTOOL_DEV_x_OPTION_x_VALUE (Page 86)
OPT_BCRELAY (Page 76)
OPT_ETHTOOL (Page 86)
OPT_IPSET (Page ??)

Obsolete variables

Package BASE

New variables

ARCH (Page ??)
COMP_TYPE_ROOTFS (Page 27)
DEBUG_IPTABLES (Page 30)
FLI4L_BUILDDATE (Page ??)
FLI4L_BUILDDIR (Page ??)
FLI4L_BUILDTIME (Page ??)
FLI4L_VERSION (Page ??)
LIBATA_DMA (Page 25)
PF_DNS_EXCEPTIONS (Page ??)
PF_INPUT_ICMP_ECHO_REQ_SIZE (Page 52)
PF_OUTPUT_ACCEPT_DEF (Page 54)
PF_OUTPUT_CT_ACCEPT_DEF (Page 66)
PF_OUTPUT_CT_N (Page 66)
PF_OUTPUT_CT_x (Page 66)
PF_OUTPUT_CT_x_COMMENT (Page 66)
PF_OUTPUT_LOG (Page 54)
PF_OUTPUT_LOG_LIMIT (Page 54)

Obsolete variables

COMPRESS_KERNEL
COMPRESS_OPT
COMPRESS_ROOTFS
DENY_ICMP
DMZ_LOG
DMZ_NAT
DMZ_ORANGE_RED_N
DMZ_ORANGE_RED_x
DMZ_ORANGE_ROUTER_N
DMZ_ORANGE_ROUTER_x
DMZ_RED_DEV
FORWARD_DENY_PORT_N
FORWARD_DENY_PORT_x
FORWARD_HOST_N
FORWARD_HOST_WHITE
FORWARD_HOST_x
IMOND_USE_ORIG

PF_OUTPUT_N (Page 54)
PF_OUTPUT_POLICY (Page 54)
PF_OUTPUT_REJ_LIMIT (Page 54)
PF_OUTPUT_UDP_REJ_LIMIT (Page 54)
PF_OUTPUT_x (Page 54)
PF_OUTPUT_x_COMMENT (Page 54)
PF_PREROUTING_CT_ACCEPT_DEF (Page 66)
PF_PREROUTING_CT_N (Page 66)
PF_PREROUTING_CT_x (Page 66)
PF_PREROUTING_CT_x_COMMENT (Page 66)
SYSLOGD_ROTATE_AT_SHUTDOWN (Page 73)

INPUT_ACCEPT_PORT_N
INPUT_ACCEPT_PORT_x
INPUT_POLICY
IP_NET_x_TYPE
MASQ_NETWORK
OPT_DMZ
OPT_EVSS
OPT_MOUNTFLOPPY
PACKETFILTER_LOG
PACKETFILTER_LOG_LEVEL
PF_NEW_CONFIG
PRESERVE
ROUTE_NETWORK
TRUSTED_NETS

Package DHCP_CLIENT

New variables

DHCP_CLIENT_x_WAIT (Page 91)

Obsolete variables

Package DNS_DHCP

New variables

DNS_AUTHORITATIVE (Page 95)
DNS_AUTHORITATIVE_IPADDR (Page 95)
DNS_AUTHORITATIVE_NS (Page 95)
DNS_BIND_INTERFACES (Page 93)
DNS_FORWARD_LOCAL (Page **??**)
DNS_LISTEN_N (Page 93)
DNS_LISTEN_x (Page 93)
DNS_LOCAL_HOST_CACHE_TTL (Page 95)
DNS_ZONE_DELEGATION_N (Page 96)
DNS_ZONE_DELEGATION_x_DOMAIN_N (Page **??**)
DNS_ZONE_DELEGATION_x_DOMAIN_x (Page **??**)
DNS_ZONE_DELEGATION_x_NETWORK_N (Page **??**)
DNS_ZONE_DELEGATION_x_NETWORK_x (Page **??**)
DNS_ZONE_DELEGATION_x_UPSTREAM_-
SERVER_N (Page **??**)
DNS_ZONE_DELEGATION_x_UPSTREAM_-
SERVER_x_IP (Page **??**)
DNS_ZONE_DELEGATION_x_UPSTREAM_-
SERVER_x_QUERYSOURCEIP (Page **??**)
DNS_ZONE_NETWORK_N (Page 96)
DNS_ZONE_NETWORK_x (Page 96)
HOST_x_IP6_NET (Page **??**)
OPT_YADIFA (Page **??**)
YADIFA_ALLOW_QUERY_N (Page 102)
YADIFA_ALLOW_QUERY_x (Page **??**)
YADIFA_LISTEN_N (Page 102)
YADIFA_LISTEN_x (Page **??**)
YADIFA_SLAVE_ZONE_N (Page 102)
YADIFA_SLAVE_ZONE_x (Page 102)
YADIFA_SLAVE_ZONE_x_ALLOW_QUERY_N
(Page 102)
YADIFA_SLAVE_ZONE_x_ALLOW_QUERY_x
(Page 102)
YADIFA_SLAVE_ZONE_x_MASTER (Page 102)
YADIFA_SLAVE_ZONE_x_USE_DNSMASQ_ZONE_-
DELEGATION (Page **??**)

Obsolete variables

DNS_LISTENIP_N
DNS_LISTENIP_x
DNS_SPECIAL_N
DNS_SPECIAL_x_DNSIP
DNS_SPECIAL_x_DOMAIN
DNS_SPECIAL_x_NETWORK

YADIFA_USE_DNSMASQ_ZONE_DELEGATION
(Page **??**)

Package DSL

New variables

FRITZDSL_FILTER_EXPR (Page 106)
PPPOE_FILTER_EXPR (Page 106)
PPTP_FILTER_EXPR (Page 106)

Obsolete variables

OPT_PFC
OPT_PPPOE_CIRC
PPPOE_CIRC_N
PPPOE_CIRC_x_CHARGEINT
PPPOE_CIRC_x_DEBUG
PPPOE_CIRC_x_ETH
PPPOE_CIRC_x_FILTER
PPPOE_CIRC_x_HUP_TIMEOUT
PPPOE_CIRC_x_MRU
PPPOE_CIRC_x_MTU
PPPOE_CIRC_x_NAME
PPPOE_CIRC_x_PASS
PPPOE_CIRC_x_TIMES
PPPOE_CIRC_x_TYPE
PPPOE_CIRC_x_USEPEERDNS
PPPOE_CIRC_x_USER

Package DYNDNS

New variables

DYNDNS_LOGINTIME (Page 114)
DYNDNS_x_LOGIN (Page 114)
OPT_STUN (Page **??**)
STUN_SERVER_N (Page **??**)
STUN_SERVER_x (Page **??**)

Obsolete variables

Package HD

New variables

Obsolete variables

HDIT_DATA
HDIT_POWEROFF
HDIT_SIZES
OPT_HDINSTALL_TEST

Package HTTPD

New variables

HTTPD_ARPING_IGNORE_N (Page 122)
HTTPD_ARPING_IGNORE_x (Page 122)

Obsolete variables

Package IPV6

New variables

IPV6_NET_x_ADVERTISE_PREF_LIFETIME (Page ??)
IPV6_NET_x_ADVERTISE_VALID_LIFETIME (Page ??)
PF6_DNS_EXCEPTIONS (Page ??)
PF6_INPUT_ICMP_ECHO_REQ_LIMIT (Page ??)
PF6_INPUT_ICMP_ECHO_REQ_SIZE (Page 138)
PF6_LOG_LEVEL (Page 137)
PF6_OUTPUT_ACCEPT_DEF (Page 141)
PF6_OUTPUT_CT_ACCEPT_DEF (Page ??)
PF6_OUTPUT_CT_N (Page ??)
PF6_OUTPUT_CT_x (Page ??)
PF6_OUTPUT_CT_x_COMMENT (Page ??)
PF6_OUTPUT_LOG (Page 141)
PF6_OUTPUT_LOG_LIMIT (Page 141)
PF6_OUTPUT_N (Page 141)
PF6_OUTPUT_POLICY (Page 140)
PF6_OUTPUT_REJ_LIMIT (Page 141)
PF6_OUTPUT_UDP_REJ_LIMIT (Page 141)
PF6_OUTPUT_x (Page 141)
PF6_OUTPUT_x_COMMENT (Page 142)
PF6_POSTROUTING_N (Page 142)
PF6_POSTROUTING_x (Page 142)
PF6_POSTROUTING_x_COMMENT (Page 142)
PF6_PREROUTING_CT_ACCEPT_DEF (Page ??)
PF6_PREROUTING_CT_N (Page ??)
PF6_PREROUTING_CT_x (Page ??)
PF6_PREROUTING_CT_x_COMMENT (Page ??)
PF6_PREROUTING_N (Page 143)
PF6_PREROUTING_x (Page 143)
PF6_PREROUTING_x_COMMENT (Page 143)

Obsolete variables

Package ISDN

New variables

ISDN_FILTER_EXPR (Page 148)
OPT_RCAPID (Page 159)
RCAPID_PORT (Page 159)
TELMOND_CAPI_CTRL_N (Page 158)
TELMOND_CAPI_CTRL_x (Page 158)

Obsolete variables

Package OPENVPN

New variables

OPENVPN_x_IPV6 (Page ??)
OPENVPN_x_LOCAL_VPN_IPV6 (Page 165)
OPENVPN_x_PF6_FORWARD_N (Page 174)
OPENVPN_x_PF6_FORWARD_x (Page 174)
OPENVPN_x_PF6_INPUT_N (Page 174)
OPENVPN_x_PF6_INPUT_x (Page 174)
OPENVPN_x_REMOTE_VPN_IPV6 (Page 165)
OPENVPN_x_RENEG_SEC (Page ??)

Obsolete variables

OPENVPN_DEFAULT_PF_DMZ_TYPE
OPENVPN_VERSION
OPENVPN_x_PF_DMZ_TYPE

Package PCMCIA

New variables

Obsolete variables

PCMCIA_CARDMGR_OPTS
PCMCIA_CORE_OPTS
PCMCIA_PCIC_EXTERN

Package PROXY

New variables

Obsolete variables

Package QOS

New variables

Obsolete variables

Package SSHD

New variables

Obsolete variables

OPT_SCP

Package TOOLS

New variables	Obsolete variables
FTP_PF_ENABLE_ACTIVE (Page 222)	OPT_ARP
OPT_ATH_INFO (Page 226)	OPT_BCRELAY
OPT_DHCPDUMP (Page 224)	OPT_ETHTOOL
OPT_DIG (Page 221)	OPT_NETSTAT
OPT_I2CTOOLS (Page 226)	OPT_SERIAL
OPT_IWLEEPROM (Page 226)	OPT_TRACEROUTE
OPT_NGREP (Page 222)	OPT_TRACEROUTE6
OPT_OPENSSL (Page 227)	WGET_SSL
OPT_REAVER (Page 227)	
OPT_SOCAT (Page 224)	

Package USB

New variables	Obsolete variables
	USB_LOWLEVEL

B.15. Differences version 3.10.5 and version 3.10.3

Package TOOLS

New variables	Obsolete variables
OPT_DHCPDUMP (Page 224)	

B.16. Differences version 3.10.5 and version 3.10.4

Package TOOLS

New variables	Obsolete variables
OPT_DHCPDUMP (Page 224)	

List of Figures

List of Tables

Index