

# **fli4l – flexible internet router for linux**

## **Version 4.0.0-trunk-x86\_64-r60727**

Frank Meyer

E-Mail: [frank@fli4l.de](mailto:frank@fli4l.de)

Das fli4l-Team

E-Mail: [team@fli4l.de](mailto:team@fli4l.de)

29. August 2022

# Inhaltsverzeichnis

<b>1. Dokumentation des Basispaketes</b>	<b>5</b>
1.1. Einleitung . . . . .	5
<b>2. Installation und Konfiguration</b>	<b>8</b>
2.1. Entpacken der Archive . . . . .	8
2.2. Konfiguration . . . . .	9
2.2.1. Editieren der Konfigurationsdateien . . . . .	9
2.2.2. Konfiguration über eine spezielle Konfigurationsdatei . . . . .	10
2.2.3. Variablen . . . . .	10
2.3. Installationsvarianten . . . . .	10
2.3.1. Router auf einem USB-Stick . . . . .	11
2.3.2. Router auf einer CD oder Netzwerkboot . . . . .	11
2.3.3. Typ A: Router auf Festplatte – nur eine FAT-Partition . . . . .	11
2.3.4. Typ B: Router auf Festplatte – je eine FAT- und ext3-Partition . . . . .	12
<b>3. Basiskonfiguration</b>	<b>13</b>
3.1. Beispiel-Datei . . . . .	14
3.2. Allgemeine Einstellungen . . . . .	23
3.3. Konsolen-Einstellungen . . . . .	30
3.4. Hilfen zum Einkreisen von Problemen und Fehlern . . . . .	31
3.5. Verwendung einer eigenen /etc/inittab . . . . .	32
3.6. Länderspezifische Tastaturlayouts . . . . .	33
3.7. Ethernet-Netzwerkkarten-Treiber . . . . .	33
3.8. Netzwerk-Konfiguration (IPv4) . . . . .	36
3.9. Netzwerk-Konfiguration (IPv6) . . . . .	38
3.9.1. Einleitung . . . . .	38
3.9.2. Adressformat . . . . .	39
3.9.3. Konfiguration . . . . .	40
3.10. Netzwerkpräfix-Konfiguration . . . . .	41
3.10.1. Netzwerkpräfixe vom Typ “stable” . . . . .	42
3.10.2. Netzwerkpräfixe vom Typ “generated-ula” . . . . .	43
3.11. Zusätzliche Routen (IPv4) . . . . .	43
3.12. Zusätzliche Routen (IPv6) . . . . .	44
3.13. Der Paketfilter (IPv4) . . . . .	44
3.13.1. Aktionen des Paketfilters . . . . .	46
3.13.2. Einschränkungen in den Regeln . . . . .	47
3.13.3. Der Einsatz von Schablonen im Paketfilter . . . . .	50
3.13.4. Die Konfiguration des Paketfilters . . . . .	54
3.13.5. Beispiele . . . . .	61
3.13.6. Standardkonfigurationen . . . . .	64

3.13.7. DMZ – Demilitarisierte Zone . . . . .	69
3.13.8. Conntrack-Helfer . . . . .	69
3.14. Der Paketfilter (IPv6) . . . . .	71
3.15. Domain-Konfiguration . . . . .	77
3.16. imond-Konfiguration . . . . .	78
3.17. Circuit-Konfiguration . . . . .	81
3.17.1. Circuits allgemein . . . . .	81
3.17.2. Circuit-Zustände . . . . .	89
3.17.3. Wählmodus (DIALMODE) . . . . .	90
3.17.4. Circuit-Klassen . . . . .	92
3.17.5. Das Programm <code>fli4lctrl</code> . . . . .	92
3.17.6. Das Programm <code>circd</code> . . . . .	95
3.17.7. Wann ist mein Router online? . . . . .	96
3.17.8. Sonstige Einstellungen . . . . .	98
3.18. Spezielle Circuit-Typen im base-Paket . . . . .	98
3.18.1. Circuits vom Typ “route” . . . . .	98
<b>4. Pakete</b>	<b>100</b>
4.1. Werkzeuge im Basispaket . . . . .	100
4.1.1. OPT_SYSLOGD – Protokollieren von Systemmeldungen . . . . .	100
4.1.2. OPT_KLOGD – Protokollieren von Kernelmeldungen . . . . .	102
4.1.3. OPT_LOGIP – Protokollieren von WAN-IP-Adressen . . . . .	102
4.1.4. OPT_Y2K – Datumskorrektur bei nicht Y2K-festen Rechnern . . . . .	102
4.1.5. OPT_PNP – Installation von isapnp tools . . . . .	103
4.1.6. OPT_HOTPLUG_PCI – Aktivieren von PCI-Hotplugging . . . . .	105
<b>5. Erzeugen der fli4l Archive/Bootmedien</b>	<b>106</b>
5.1. Erzeugen der fli4l Archive/Bootmedien unter Linux bzw. anderen Unix-Derivaten und Mac OS X . . . . .	106
5.1.1. Kommandozeilenoptionen . . . . .	107
5.2. Erzeugen der fli4l Archive/Bootmedien unter Windows . . . . .	109
5.2.1. Kommandozeilenoptionen . . . . .	109
5.2.2. Konfigurationsdialog – Einstellung des Konfigurationsverzeichnis . . . . .	110
5.2.3. Konfigurationsdialog – allgemeine Einstellungen . . . . .	111
5.2.4. Konfigurationsdialog – Einstellungen für Remoteupdate . . . . .	112
5.2.5. Konfigurationsdialog – Einstellungen für HD-pre-install . . . . .	113
5.3. Steuerungsdatei <code>mkfli4l.txt</code> . . . . .	114
<b>6. Anbindung von PCs im LAN</b>	<b>116</b>
6.1. IP-Adresse . . . . .	116
6.2. Rechnername und Domain . . . . .	116
6.2.1. Windows 2000 . . . . .	116
6.2.2. NT 4.0 . . . . .	117
6.2.3. Win95/98 . . . . .	117
6.2.4. Windows XP . . . . .	117
6.2.5. Windows 7 . . . . .	118
6.2.6. Windows 8 . . . . .	118

6.3. Gateway . . . . .	118
6.4. DNS-Server . . . . .	119
6.5. Verschiedenes . . . . .	119
<b>7. Client-/Server-Schnittstelle imon</b>	<b>120</b>
7.1. imon-Server imond . . . . .	120
7.1.1. Least-Cost-Routing – Funktionsweise . . . . .	120
7.1.2. Zur Berechnung der Onlinekosten . . . . .	125
7.2. Windows-Client imonc.exe . . . . .	125
7.2.1. Einleitung . . . . .	125
7.2.2. Startparameter . . . . .	126
7.2.3. Seite Überblick . . . . .	128
7.2.4. Config-Dialog . . . . .	129
7.2.5. Seite Anrufe . . . . .	135
7.2.6. Seite Verbindungen . . . . .	135
7.2.7. Seite Fax . . . . .	136
7.2.8. Seite E-Mail . . . . .	136
7.2.9. Admin . . . . .	137
7.2.10. Seiten Fehler, Syslog und Firewall . . . . .	138
7.2.11. Seite News . . . . .	138
7.3. Unix/Linux-Client imonc . . . . .	138
<b>A. Anhang zum Basispaket</b>	<b>141</b>
A.1. Nullmodemkabel . . . . .	141
A.2. Serielle Konsole . . . . .	141
A.3. Programme . . . . .	142
A.4. Andere i4l-Tools . . . . .	142
A.5. Fehlersuche . . . . .	142
A.6. Literaturhinweise . . . . .	143
A.7. Präfixe . . . . .	144
A.8. Gewähr und Haftung . . . . .	144
A.9. Danke . . . . .	144
A.9.1. Projektgründung . . . . .	144
A.9.2. Entwickler- und Testteam . . . . .	144
A.9.3. Entwickler- und Testteam (nicht mehr aktive) . . . . .	145
A.9.4. Sponsoren . . . . .	146
A.10.Feedback . . . . .	147
<b>Abbildungsverzeichnis</b>	<b>148</b>
<b>Tabellenverzeichnis</b>	<b>149</b>
<b>Index</b>	<b>150</b>

# 1. Dokumentation des Basispaketes

## 1.1. Einleitung

fli4l ist ein auf Linux basierender ISDN-, DSL-, UMTS- und Ethernet-Router mit geringen Anforderungen an die zugrunde liegende Hardware: Ein USB-Stick als Bootmedium, ein Intel Pentium MMX-Prozessor, 64 MiB RAM sowie (mindestens) eine Ethernet-Netzwerkkarte sind dafür vollkommen ausreichend. Das notwendige Bootmedium kann unter Linux, Mac OS X oder MS Windows erstellt werden. Dabei sind keine Linux-Kenntnisse erforderlich, aber durchaus hilfreich. Grundkenntnisse von Netzwerken, TCP/IP, DNS und Routing sollten jedoch vorhanden sein. Für eigene Erweiterungen/Entwicklungen, welche über die Standardkonfiguration hinausgehen, sind ein lauffähiges Linux-System und Linux-Kenntnisse notwendig.

fli4l unterstützt verschiedene Bootmedien, darunter USB-Sticks, Festplatten, CDs und nicht zuletzt das Booten über das Netzwerk. Ein USB-Stick ist in vielerlei Hinsicht ideal: Heutzutage kann so gut wie jeder PC von einem USB-Stick starten, er ist recht erschwinglich, er hat eine ausreichende Größe, und man kann sowohl unter Linux als auch unter MS Windows auf relativ einfache Weise eine fli4l-Installation darauf ablegen. Auch ist er im Gegensatz zu einer CD beschreibbar und kann somit nichtflüchtige Konfigurationsdaten (wie z.B. DHCP-Leases) speichern.

- Allgemeine Features
  - Erstellen von Bootmedien unter [Linux](#) (Seite 106), [Mac OS X](#) (Seite 106) und [MS Windows](#) (Seite 109)
  - Konfiguration über normale ASCII/UTF-8-Dateien
  - Unterstützung von IP-Masquerading und Portweiterleitung
  - Least-Cost-Routing (LCR): automatische Auswahl des Providers, je nach Uhrzeit
  - Anzeige/Berechnung/Protokollierung von Verbindungszeiten und -kosten
  - MS Windows/Linux-Client imonc mit Schnittstelle zu imond und telmond
  - Upload von aktualisierten Konfigurationsdateien über MS Windows-Client imonc oder via SCP unter Linux
  - Bootmedien nutzen das VFAT-Dateisystem zum dauerhaften Speichern von Dateien
  - Paketfilter: Protokollieren von Zugriffen von außen auf gesperrte Ports
  - Einheitliche Abbildung von WAN-Schnittstellen auf sogenannte Circuits
  - Paralleler Betrieb von ISDN- und DSL/UMTS-Circuits ist möglich
- Router-Basisfunktionalität
  - Linux-Kernel 3.18 oder 3.19
  - Paketfilter und IP-Masquerading

## 1. Dokumentation des Basispaketes

- DNS-Server, um die Anzahl von DNS-Abfragen an externe DNS-Server zu reduzieren
- Netzwerkfähiger imond-Server mit Monitor- und LCR-Steuerfunktionen
- Netzwerkfähiger telmond-Server zur Protokollierung von eingehenden Telefonanrufen
- Ethernet-Unterstützung
  - Aktuelle Netzwerkkartentreiber: Unterstützung von über 140 Kartentypen
- DSL-Unterstützung
  - Roaring Penguin PPPoE-Treiber, mit Dial-on-Demand (abschaltbar)
  - PPTP für DSL-Anbindungen in Österreich und den Niederlanden
- ISDN-Unterstützung
  - Unterstützung von knapp 60 ISDN-Kartentypen
  - Mehrere ISDN-Verbindungsmöglichkeiten: eingehend/ausgehend/Rückruf, „roh“/Punkt-zu-Punkt (ppp)
  - Kanalbündelung: automatische Bandbreitenanpassung oder manuelle Zuschaltung des zweiten Kanals über MS Windows-/Linux-Client
- Optionale Programmpakete
  - DNS-Server
  - DHCP-Server
  - SSH-Server
  - Einfache Online/Offline-Anzeige über LED
  - Serielle Konsole
  - Mini-Webserver für ISDN- und DSL-Monitoring sowie zur Rekonfiguration und/oder Aktualisierung des Routers
  - Zugangserlaubnis für bestimmte konfigurierte Netzwerke von außen
  - Unterstützung für PCMCIA-Karten (heutzutage PC-Cards genannt)
  - Protokollierung von Systemmeldungen
  - Konfiguration von ISAPnP-Karten mit den isapnp-Werkzeugen
  - Zusätzliche Werkzeuge zum Debugging
  - Konfiguration der seriellen Schnittstelle
  - Notfallsystem zur Fernwartung über ISDN
  - Software zur Anzeige konfigurierbarer Informationen auf einem LCD, z.B. von Übertragungsraten, CPU-Auslastung etc.
  - PPP-Server/Router über serielle Schnittstelle
  - ISDN-Modem-Emulator über serielle Schnittstelle

## 1. Dokumentation des Basispaketes

- Druckerserver
  - Synchronisierung der Uhrzeit mit externen Zeit-Servers
  - Ausführen von benutzerdefinierten Kommandos bei eingehenden Telefonanrufen (z.B. um ein Internet-Einwahl durchzuführen)
  - Unterstützung von IP-Aliasing (mehrere IP-Adressen pro Netzwerkschnittstelle)
  - VPN-Unterstützung
  - IPv6-Unterstützung
  - WLAN-Unterstützung: fli4l kann sowohl Zugangsknoten als auch Client sein
  - RRD-Tool zum Überwachen des fli4l
  - und vieles andere mehr...
- Hardwarevoraussetzungen
    - Intel Pentium-Prozessor mit MMX Unterstützung
    - 64 MiB Speicher, besser 128 MiB
    - Ethernet-Netzkarte
    - ISDN: unterstützte ISDN-Karte
    - ein USB-Stick, eine ATA-Festplatte oder eine CF-Karte (die genauso wie eine ATA-Festplatte angesprochen wird); alternativ ist auch der Start von CD möglich
  - Softwarevoraussetzungen

Unter Linux werden folgende Programme vorausgesetzt:

    - GCC und GNU make
    - syslinux
    - mtools (mcopy)

Unter MS Windows werden keine zusätzlichen Werkzeuge benötigt, fli4l bringt alles Notwendige mit.

Zusätzlich gibt es zur Steuerung/Statusanzeige des fli4l-Routers noch den Client imonc. Dieses Programm ist für MS Windows (windows/imonc.exe) und auch für Linux (unix/gtk-imonc) vorhanden.

Und nun ...

Viel Spaß mit fli4l!

Frank Meyer und das fli4l-Team

E-Mail: [team@fli4l.de](mailto:team@fli4l.de)

## 2. Installation und Konfiguration

### 2.1. Entpacken der Archive

Unter Linux:

```
tar xvfz fli4l-4.0.0-trunk-x86_64-r60727.tar.gz
```

Funktioniert dies nicht, geht's auch so:

```
gzip -d < fli4l-4.0.0-trunk-x86_64-r60727.tar.gz | tar xvf -
```

Wer die aktuelle Version in einem bereits existierenden fli4l-Verzeichnis installiert, sollte anschließend `mkfli4l.sh -c` aufrufen, also:

```
cd fli4l-4.0.0-trunk-x86_64-r60727
sh mkfli4l.sh -c
```

Es wird jedoch empfohlen, ein neues Verzeichnis für eine neue Version zu benutzen – die Konfiguration kann durch ein entsprechendes Werkzeug zum Dateivergleich sehr einfach übernommen werden.

Unter MS Windows kann das komprimierte Tar-Archiv zum Beispiel mit WinZip extrahiert werden. Dabei ist jedoch zu beachten, dass die Dateien *mit* Unterverzeichnissen (Einstellung in WinZip überprüfen!) ausgepackt werden. Außerdem ist in *Optionen*  $\Rightarrow$  *Konfiguration* die so genannte „Smart TAR CR conversion“ abzuschalten. Ist diese eingeschaltet, werden einige wichtige Dateien von WinZip falsch extrahiert.

Alternativ ist das OpenSource-Programm 7-Zip (<http://www.7-zip.org/>) sehr zu empfehlen, welches ebenso mächtig wie WinZip ist.

Es werden folgende Dateien im Unterverzeichnis `fli4l-4.0.0-trunk-x86_64-r60727/` installiert:

- Dokumentation:
  - `doc/deutsch/*` Deutsche Dokumentation
  - `doc/english/*` Englische Dokumentation
  - `doc/french/*` Französische Dokumentation
- Konfiguration:
  - `config/*.txt` Konfigurationsdateien, diese müssen bearbeitet werden
- Skripte/Prozeduren:
  - `mkfli4l.sh` Boot-Medium oder Dateien erzeugen: Linux/Unix-Version
  - `mkfli4l.bat` Boot-Medium erzeugen: Windows-Version



- Kernel/Boot-Dateien:
  - img/kernel Linux-Kernel
  - img/boot\*.msg Bootscreen Texte
- Zusatzpakete:
  - opt/\*.txt Diese Dateien beschreiben, was bei welchen Einstellungen in das Archiv opt.img gelangt.
  - opt/... Optionale Kernel-Module, Dateien und Programme
- Quellcode:
  - src/\* Quellcode/Werkzeuge für Linux, siehe src/README
- Programme:
  - unix/mkfli4l\* Erzeugen des Bootmediums: Unix/Linux-Version
  - windows/\* Erzeugen des Bootmediums: Windows-Version
  - unix/imonc\* imond-Client für Unix/Linux
  - windows/imonc/\* imond-Client für Windows

## 2.2. Konfiguration

### 2.2.1. Editieren der Konfigurationsdateien

Zur Konfiguration von fli4l müssen lediglich die Dateien config/\*.txt angepasst werden. Um im Nachhinein die eigenen Konfiguration mit der ausgelieferten vergleichen zu können oder um mehrere Konfigurationen verwalten zu können, empfiehlt es sich, eine Kopie des config-Verzeichnisses anzulegen und die Konfiguration in dieser Kopie durchzuführen. Ein Vergleich der Konfigurationen ist dann durch Verwendung eines geeigneten Werkzeugs (z.B. “diff” unter \*nix) relativ einfach möglich. Nehmen wir einmal an, die eigene config liegt in einem Verzeichnis mit Namen “meine\_config” ebenfalls im fli4l-Verzeichnis dann wäre der Aufruf wie folgt:

```
~/src/fli4l> diff -u {config,meine_config}/build/rc.cfg | grep '^[+-]'
```

---	config/build/rc.cfg	2014-02-18 15:34:39.085103706	+0100
+++	meine_config/build/rc.cfg	2014-02-18 15:34:31.094317441	+0100

```
-PASSWORD='/P6h4i0IN5Bbc'  
+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='anonymer'  
-PPPOE_PASS='surfer'  
+PPPOE_USER='ich'  
+PPPOE_PASS='mein-passwd'  
-OPT_SSHD='no'  
+OPT_SSHD='yes'
```

## 2. Installation und Konfiguration

Man sieht hier auch sehr schön, dass ein einfacher DSL-Router mit wenigen Handgriffen konfiguriert ist, auch wenn einen die Konfigurationsdateien auf den ersten Blick mit ihrer Fülle von Einstellungsmöglichkeiten erschlagen.

### 2.2.2. Konfiguration über eine spezielle Konfigurationsdatei

Da sich die Konfiguration durch das Modul-Konzept auf verschiedene Dateien verteilt, und das Bearbeiten dadurch unter Umständen etwas mühsam wird, kann man die Konfiguration auch in einer einzelnen Datei namens `<configverzeichnis>/_fli4l.txt` ablegen, deren Inhalt dann zusätzlich zu den normalen Konfigurationsdateien eingelesen wird und deren Inhalt dominiert. Um beim obigen Beispiel zu bleiben: Um einen einfachen DSL-Router zu konfigurieren, könnten wir einfach folgendes in diese Datei schreiben:

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

Man sollte vermeiden, beide Konfigurationsvarianten zu mischen.

### 2.2.3. Variablen

Sie werden merken, dass einige Variablen auskommentiert sind. Wenn das der Fall ist, erhält sie eine sinnvolle Standard-Belegung. Diese Standard-Belegung ist für jede Variable dokumentiert. Wünschen Sie einen anderen Wert für diese Variable, sollten Sie das Kommentarzeichen am Anfang der Variablendefinition (`'#'`) entfernen und den entsprechenden Wert zwischen den Hochkommata einfügen.

## 2.3. Installationsvarianten

In den vorhergehenden Versionen von `fli4l` wurde lediglich das Booten von einer Diskette unterstützt. Dies ist aus oben genannten Gründen nun nicht mehr möglich, aber die Alternative mittels eines USB-Sticks ist gegeben.

Es sind auch eine Vielzahl anderer Bootmedien (CD, HD, Netzwerk, Compact-Flash, DoC, ...) möglich und `fli4l` kann auch auf diversen Medien installiert (HD, Compact-Flash, DoC) werden. Dazu kann `fli4l` auf drei verschiedenen Wegen gebootet werden:

**Single Image** Der Bootloader lädt den Linux-Kern und dann `fli4l` als ein einziges Image — danach kann `fli4l` ohne weiteren Zugriff auf andere Medien booten. Beispiele dafür sind die Boottypen *integrated*, *attached*, *netboot* und *cd*.

**Split Image** Der Bootloader lädt den Linux-Kern und dann ein rudimentäres `fli4l`-Image, dass die Bootmedien einbindet und die Konfiguration und restlichen Dateien aus einem dort liegenden Archiv holt. Beispiele dafür sind diese Boottypen: *hd (Typ A)*, *ls120*, *attached* und *cd-emul*.

**Installation auf einem Medium** Der Bootloader lädt den Linux-Kern und dann ein rudimentäres fli4l-Image, das eine bereits vorhandene fli4l-Installation in sein Dateisystem einbindet und damit keine weiteren Archive auspacken muss. Eine HD-Installation vom Typ B ist ein Beispiel dafür.

Man sollte jedoch zunächst erst einmal fli4l in einer minimalen Version installieren und damit Erfahrungen sammeln. Möchte man später fli4l zusätzlich als Anrufbeantworter und als HTTP-Proxy einsetzen, so hat man vorher schon mal Erfahrungen mit einem grundsätzlich laufenden Router.

Für die Installation ergeben sich daraus die folgenden fünf Varianten:

**USB-Stick** Router auf einem USB-Stick

**CD-router** Router auf einer CD

**Netzwerk** Netzwerkboot

**HD-Installation Typ A** Router auf Festplatte, CF, DoC – nur eine FAT-Partition

**HD-Installation Typ B** Router auf Festplatte, CF, DoC – je eine FAT- und ext3-Partition

### 2.3.1. Router auf einem USB-Stick

USB-Sticks werden von Linux als Festplatten angesprochen, daher gelten hier die Ausführungen zur Festplatteninstallation entsprechend. Bitte beachten Sie, dass mittels des `OPT_USB` die entsprechenden Treiber geladen werden müssen, damit der Stick mittels `OPT_HDINSTALL` eingebunden werden kann.

### 2.3.2. Router auf einer CD oder Netzwerkboot

Alle benötigten Dateien liegen auf dem Bootmedium und werden beim Booten in eine dynamische RAM-Disk entpackt. In einer Minimalkonfiguration ist damit ein Betrieb des Routers mit nur 64 MiB RAM möglich. Die maximale Konfiguration wird nur durch die Kapazität des Bootmediums und des Hauptspeichers limitiert.

### 2.3.3. Typ A: Router auf Festplatte – nur eine FAT-Partition

Dies entspricht der CD-version, nur dass die Dateien hierbei auf einer Festplatte liegen, wobei der Begriff „Festplatte“ hier auch Compact-Flash-Medien ab 8 MiB und andere Geräte, welche Linux als Festplatte ansprechen kann, mit einschließt. Seit fli4l 2.1.4 können auch DiskOnChip Flash-Speicher von M-Sys oder SCSI-Festplatten benutzt werden.

Die Beschränkung des Archivs `opt.img` durch die Diskettenkapazität wird aufgehoben, aber alle diese Dateien müssen in einer RAM-Disk mit der entsprechenden Größe beim Boot installiert werden. Dies erhöht den RAM-Bedarf beim Einsatz vieler Pakete.

Für ein Update der Softwarepakete (d.h. des Archivs `opt.img` und der `rc.cfg` über das Netzwerk) muss die FAT-Partition genügend Platz für den Kernel, das RootFS und die DOPPELTE Größe des `opt.img` haben! Falls auch die Notfall-Option genutzt werden soll, erhöht sich der Platzbedarf noch einmal um die Größe des `opt.img`.

#### **2.3.4. Typ B: Router auf Festplatte – je eine FAT- und ext3-Partition**

Im Gegensatz zum Typ A werden hier nicht alle Dateien in die Ramdisk gepackt, sondern bei dem erstmaligen Start nach der Installation oder nach einem Update aus dem Archiv `opt.img` direkt auf eine ext3-Partition kopiert und im späteren Betrieb von dort geladen. Bei dieser Version ist der Speicherbedarf für die RAM-Disk am geringsten und damit meist auch ein Betrieb mit sehr wenig RAM möglich.

Weitere Informationen zur Installation auf Festplatten finden Sie in der Dokumentation des Pakets `HD` (separat herunterzuladen)- beginnend bei der Beschreibung der Variablen `OPT_HDINSTALL`.

### 3. Basiskonfiguration

Ab Version 2.0 ist die fli4l-Distribution modular aufgebaut und in mehrere Pakete aufgeteilt, die extra heruntergeladen werden müssen. Im Paket `fli4l-4.0.0-trunk-x86_64-r60727.tar.gz` ist lediglich die Basis-Software für einen Ethernet-Router enthalten. Für DSL, ISDN und weitere Software müssen die Pakete separat heruntergeladen werden und ausgehend vom Verzeichnis `fli4l-4.0.0-trunk-x86_64-r60727/` (!) installiert werden. Durch die Auswahlmöglichkeit des Betriebssystemkerns von fli4l sind diese in die Kernel Pakete ausgelagert worden. Somit ist als Minimum Basis und ein Kernel Paket erforderlich. In Tabelle 3.1 finden Sie einen Überblick über die Zusatzpakete.

Die zur Konfiguration des fli4l-Routers verwendeten Dateien befinden sich im Verzeichnis `config/` und werden hier im Folgenden beschrieben.

Diese Dateien können mit einem *einfachen* Text-Editor oder auch mit einem speziell an fli4l angepassten Editor verändert werden. Diverse Editoren sind unter

<http://www.fli4l.de/download/zusatzpakete/addons/> zu finden.

Sind spezielle Anpassungen/Erweiterungen erforderlich, die über die unten aufgeführten Einstellungsmöglichkeiten hinausgehen, benötigt man ein lauffähiges Linux-System, um Anpassungen im RootFS vorzunehmen. In diesem Fall hilft `src/README` weiter.

Tabelle 3.1.: Übersicht über die (Zusatz-)Pakete

Download-Archiv	Paket
fli4l-4.0.0-trunk-x86_64-r60727	BASIS, erforderlich!
kernel_4_19	Linux-Kernel, erforderlich!
fli4l-4.0.0-trunk-x86_64-r60727-doc	Komplette Dokumentation
advanced_networking	Erweiterte Netzwerkkonfiguration
cert	Zertifikatsverwaltung
chrony	Time-Server/Client
dhcp_client	Verschiedene DHCP-Clients
dns_dhcp	DNS- und DHCP-Server
dslmodem	Unterstützung für interne DSL-Modems (z.B. AVM Fritz!DSL)
dyndns	Unterstützung von DYNDNS-Diensten
easycron	Zeitplandienst
hd	Installation auf Festplatte
httpd	Mini-Webserver für Status-Ausgaben
hwsupp	Unterstützung von Hardware
imonc_windows	Der Windows-Imonc
imonc_unix	Der GTK-Unix-Imonc
ipv6	Internet Protokoll Version 6
isdn	ISDN-Router
openvpn	OpenVPN-Unterstützung
pcmcia	Unterstützung von PCMCIA-Karten
ppp	PPP-Basispaket
pppoe	DSL-Router (PPPoE)
proxy	Proxy-Server
qos	Quality of Service
sshd	SSH-Server
tools	Diverse Linux-Werkzeuge
umts	Anbindung mittels UMTS an das Internet
usb	Unterstützung der USB-Schnittstelle
vpn	VPN (PPTP)
wlan	Unterstützung von WLAN-Karten

## 3.1. Beispiel-Datei

Die Beispiel-Datei `base.txt` im Verzeichnis `config/` hat folgenden Inhalt:

```
##-----
## 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 !
##
##-----
```

### 3. Basiskonfiguration

```
## Creation:      26.06.2001  fm
## Last Update:  $Id: base.txt 60727 2022-08-29 10:29:50Z florian $
##
## 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
RTC_SYNC='hwclock'        # how to synchronize the hardware clock?
KERNEL_VERSION='5.4.211'  # 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
```

### 3. Basiskonfiguration

```
# 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[]='ne2k-pci' # 1st driver: name (e.g. NE2000 PCI clone)
#{
# OPTION='' # 1st driver: additional option
#}
#NET_DRV[]='ne' # 2nd driver: name (e.g. NE2000 ISA clone)
#{
# OPTION='io=0x320' # 2nd driver: additional option
#}

#-----
# Network prefixes
#-----
#OPT_NET_PREFIX='no' # enable use of network prefixes: yes or no
#NET_PREFIX # network prefixes not bound to an interface
#{
# [] # network prefix assignment
# {
# NAME="site" # name of network prefix
# TYPE="static" # type of network prefix
# STATIC_IPv4="192.168.10.0/24" # static IPv4 prefix
# STATIC_IPv6="fd6e:d748:fd6d::/48" # static IPv6 prefix
```



### 3. Basiskonfiguration

```
# }
#}

#-----
# ULA prefixes
#-----
#OPT_NET_PREFIX_ULA='no'          # enable generation of ULAs: yes or no
#NET_PREFIX
#{
# []
# {
#   NAME="LAN"                    # name of network prefix
#   TYPE="generated-ula"          # type of network prefix
#   ULA_DEV='eth0'                # Ethernet interface of which the MAC is taken
# }
#}

#-----
# Networks
#-----
OPT_IPV4='yes'                    # enable IPv4 networking
                                  # WARNING: Don't set this to 'no', this is
                                  # currently not supported!

#IP_NET[1]='192.168.6.1/24'        # IP address of your n'th ethernet card and
                                  # netmask in CIDR (no. of set bits)

#{
#   DEV='eth0'                    # required: device name like ethX
#}

#OPT_IPV6='no'                    # set to 'yes' to activate IPv6 support

#IPV6_NET[1]='{internet-v6}+::1:0:0:0:1/64'
                                  # The router address and net mask of
                                  # this subnet. If this subnet is associated
                                  # with a circuit (i.e. the address is
                                  # prefixed by {<circuit>}), use an address
                                  # WITHOUT the subnet prefix; when the
                                  # associated circuit comes up, its prefix
                                  # will be combined with the address
                                  # specified here to yield a complete
                                  # address.
                                  #
                                  # NOTE that the net mask must be equal to
                                  # 64 if you want to use stateless IPv6
                                  # autoconfiguration!
                                  #
                                  # In this example, a /48 subnet prefix is
                                  # assumed which is extended by the subnet
                                  # '1' and the host part '0:0:0:1'. So with
                                  # e.g. '2001:db8:13bc/48' as subnet prefix
                                  # provided by circuit 'internet-v6', the
                                  # complete address and mask becomes
```

### 3. Basiskonfiguration

```
# '2001:db8:13bc:1::1/64'.
#
# If no circuit prefix is used, no circuit
# is associated, so the address
# specification is taken "as is" and is not
# completed by any prefix

#{
#  DEV='IP_NET_1_DEV'          # interface this subnet is bound to
#  ADVERTISE='yes'            # should the subnet prefix be advertised
#                             # automatically via RA in order to enable
#                             # stateless autoconfiguration?
#  ADVERTISE_DNS='no'         # should the DNS service be advertised
#                             # within this subnet via RA?
#}

#-----
# Additional routes, optional
#-----
#IP_ROUTE[]='192.168.7.0/24 192.168.6.99'
#                             # network/netmaskbits gateway
#IP_ROUTE[]='0.0.0.0/0 192.168.6.99'
#                             # example for default-route

#IPV6_ROUTE[]='2001:db8:13bc:2::/64 2001:db8:900:530::1'
#                             # example route

#-----
# Packet filter configuration
#-----
#-----
# INPUT chain
#-----
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[]='IP_NET_1 ACCEPT'  # allow all hosts in the local network to
#                             # access the router
#PF_INPUT[]='tmpl:samba DROP NOLOG'
#                             # drop (or reject) samba access

#{
#  COMMENT='no samba traffic allowed'
#                             # without logging, otherwise the log file will
#                             # be filled with useless entries
#}

PF6_INPUT_POLICY='REJECT'     # be nice and use reject as policy
```

### 3. Basiskonfiguration

```
PF6_INPUT_ACCEPT_DEF='yes'      # use default rule set
PF6_INPUT_LOG='no'              # don't log anything
PF6_INPUT_LOG_LIMIT='3/minute:5'
                                # log 3 events per minute; allow a burst of 5
                                # events
PF6_INPUT_REJ_LIMIT='1/second:5'
                                # reject 1 connection per second; allow a burst
                                # of 5 events; otherwise drop packet
PF6_INPUT_UDP_REJ_LIMIT='1/second:5'
                                # reject 1 udp packet per second; allow a burst
                                # of 5 events; otherwise drop packet

#PF6_INPUT[]='fe80::0/10] ACCEPT'
                                # allow all hosts in the local network to
                                # access the router
#PF6_INPUT[]='IPV6_NET_1 ACCEPT'
                                # allow all hosts in the first subnet to access
                                # the router
#PF6_INPUT[]='tmpl:samba DROP NOLOG'
                                # drop (or reject) samba access

#{
# COMMENT='no samba traffic allowed'
                                # without logging, otherwise the log file will
                                # be filled with useless entries
#}

#-----
# FORWARD chain
#-----
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[]='tmpl:samba DROP' # drop samba traffic if it tries to leave the
                                # subnet
#PF_FORWARD[]='IP_NET_1 ACCEPT' # accept everything else

PF6_FORWARD_POLICY='REJECT'     # be nice and use reject as policy
PF6_FORWARD_ACCEPT_DEF='yes'    # use default rule set
PF6_FORWARD_LOG='no'            # don't log anything
PF6_FORWARD_LOG_LIMIT='3/minute:5'
                                # log 3 events per minute; allow a burst of 5
                                # events
PF6_FORWARD_REJ_LIMIT='1/second:5'
                                # reject 1 connection per second; allow a burst
```

### 3. Basiskonfiguration

```
# of 5 events; otherwise drop packet
PF6_FORWARD_UDP_REJ_LIMIT='1/second:5'
# reject 1 udp packet per second; allow a burst
# of 5 events; otherwise drop packet

#PF6_FORWARD[]='tmp1:samba DROP'
# drop samba traffic if it tries to leave the
# subnet
#PF6_FORWARD[]='IPV6_NET_1 ACCEPT'
# accept everything else

#-----
# OUTPUT chain
#-----
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[]='any 217.197.80.132 REJECT'
# don't allow the fli4l to reach fli4l.de

PF6_OUTPUT_POLICY='ACCEPT'     # default policy for outgoing packets
PF6_OUTPUT_ACCEPT_DEF='yes'    # use default rule set
PF6_OUTPUT_LOG='no'            # don't log anything
PF6_OUTPUT_LOG_LIMIT='3/minute:5'
# log 3 events per minute; allow a burst of 5
# events
PF6_OUTPUT_REJ_LIMIT='1/second:5'
# reject 1 connection per second; allow a burst
# of 5 events; otherwise drop packet
PF6_OUTPUT_UDP_REJ_LIMIT='1/second:5'
# reject 1 udp packet per second; allow a burst
# of 5 events; otherwise drop packet
#PF6_OUTPUT[]='any 2001:bf0:c000:a::2:132 REJECT'
# don't allow the fli4l to reach fli4l.de

#-----
# POSTROUTING chain
#-----
#PF_POSTROUTING[]='IP_NET_1 MASQUERADE'
# masquerade traffic leaving the subnet

#PF6_POSTROUTING[]='IPV6_NET_1 MASQUERADE'
# masquerade traffic leaving the subnet
```

### 3. Basiskonfiguration

```
#-----
# PREROUTING chain
#-----
#PF_PREROUTING[]='1.2.3.4 dynamic:22 DNAT:@client2'
# forward ssh connections coming from 1.2.3.4
# to client2

#PF6_PREROUTING[]='tmpl:ssh [2001:db8::1] DNAT:@client2'
# forward ssh connections coming from
# [2001:db8::1] to client2

#-----
# PREROUTING_CT chain
#-----
PF_PREROUTING_CT_ACCEPT_DEF='yes'
# use default rule set
#PF_PREROUTING_CT[]='tmpl:ftp IP_NET_1 HELPER:ftp'
# associate FTP conntrack helper for active FTP
# forwarded from within the LAN to some FTP
# server outside
#PF_PREROUTING_CT[]='tmpl:ftp any dynamic HELPER:ftp'
# associate FTP conntrack helper for passive
# FTP forwarded to the router's external IP
# (some PREROUTING rule needs to forward the
# packets to some FTP server within the LAN)

#PF6_PREROUTING_CT[]='tmpl:ftp IPV6_NET_1 HELPER:ftp'
# associate FTP conntrack helper for active FTP
# forwarded from within the LAN to some FTP
# server outside
#PF6_PREROUTING_CT[]='tmpl:ftp any IPV6_NET_1 HELPER:ftp'
# associate FTP conntrack helper for passive
# FTP forwarded to some FTP server within the
# LAN

#-----
# OUTPUT_CT chain
#-----
PF_OUTPUT_CT_ACCEPT_DEF='yes' # use default rule set
#PF_OUTPUT_CT[]='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')

#PF6_OUTPUT_CT[]='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')

#-----
# USER chain
```

### 3. Basiskonfiguration

```
#-----
#PF_USR_CHAIN[]='...'          # some user-defined rule
#PF6_USR_CHAIN[]='...'         # some user-defined rule

#-----
# 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_IP6='IPV6_NET_1_IPADDR'
                                # optional, can be used to explicitly set
                                # the router's IPv6 address; if left empty,
                                # this setting is taken from the first
                                # configured /64 IPv6 subnet (see below)
#HOSTNAME_ALIAS[]='router.lan.fli4l'
                                # first ALIAS name
#HOSTNAME_ALIAS[]='gateway.my.lan'
                                # second ALIAS name

#-----
# optional package: syslogd
#-----
#OPT_SYSLOGD='no'              # start syslogd: yes or no
#SYSLOGD_RECEIVER='yes'        # receive messages from network
#SYSLOGD_DEST[]='*. * /dev/console'
                                # n'th prio & destination of syslog msgs
#SYSLOGD_DEST[]='*. * @192.168.6.2'
                                # example: loghost 192.168.6.2
#SYSLOGD_DEST[]='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

#-----
```

### 3. Basiskonfiguration

```
# 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

#-----
# Optional: PCI hotplugging
#-----
#OPT_HOTPLUG_PCI='no'        # if yes, various PCI hotplugging drivers are
                             # loaded at boot time; note that ACPI hot-
                             # plugging (as used by e.g. KVM) is built into
                             # the kernel and does _not_ require this OPT to
                             # be enabled (but it doesn't hurt neither)

#-----
# Optional package: lua
# (Note: This package will eventually be integrated into the base package as
# it is planned to implement core fli4l services in Lua!)
#-----
#OPT_LUA='no'                # enable Lua

#-----
# Optional package: luatests
#-----
#OPT_LUATESTS='no'          # enable Lua test suite
#LUATESTS_RUNATBOOTTIME='yes' # set to 'yes' if test suite should run when
                             # the fli4l boots
```

Zu beachten ist, dass diese Datei im DOS-Format gespeichert ist. Das heißt, sie enthält jeweils am Zeilenende ein zusätzliches Carriage-Return (CR). Da die meisten Unix-Editoren damit keine Probleme bekommen wurde dieses Format gewählt, denn umgekehrt hat der Windows-Editor bei fehlendem CR am Zeilenende keine Chance!

Sollte es wider Erwarten unter Unix/Linux doch Probleme mit dem Lieblingseditor geben, kann die Datei vor dem Editieren mit einem Befehl in das Unix-Format konvertiert werden:

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

Für die Erstellung des Boot-Mediums ist es völlig unerheblich, ob die Datei CRs am Zeilenende enthält oder nicht. Sie werden beim Schreiben auf das Boot-Medium einschließlich der Kommentare komplett ignoriert.

Jetzt aber zum Inhalt ...

## 3.2. Allgemeine Einstellungen

**HOSTNAME** Standardwert: `HOSTNAME='fli4l'`

Als erstes sollte man seinem fli4l-Router einen Namen geben.

**PASSWORD** Standardwert: `PASSWORD='fli4l'`

Das hier angegebene Passwort wird für das Einloggen in den fli4l-Rechner benötigt – sei es per Tastatur direkt am Router oder per SSH von einem anderen Rechner aus (hierzu wird das `sshd`-Paket benötigt). Es muss aus mindestens einem und darf aus höchstens 126 Zeichen bestehen.

**BOOT\_TYPE** Standardwert: `BOOT_TYPE='hd'`

`BOOT_TYPE` legt im weitesten Sinne das Bootmedium fest. Diese Variable steuert, welche zusätzlichen Treiber (Kernel-Module) und Start-Skripte mit in das RootFS aufgenommen werden. Zum Verständnis eine kurze Skizze des Bootvorgangs:

- Das BIOS des Rechners lädt/startet den Bootloader auf dem Bootmedium.
- Der Bootloader (i.d.R. `syslinux`) entpackt, lädt und startet den Kernel.
- Der Kernel entpackt das RootFS (= das grundlegende Dateisystem mit darin enthaltenen Programmen und Skripten), mountet das RootFS und beginnt die Start-Skripte abzuarbeiten.
- Je nach `BOOT_TYPE` werden nun die Kernel-Module für das jeweilige Bootmedium geladen, die Boot-Partition gemountet und das OPT-Archiv (`opt.img`) mit den zusätzlichen Programmen entpackt.
- Im Anschluss beginnt die Konfiguration der einzelnen Dienste des fli4l.

Zur Zeit sind folgende Werte für `BOOT_TYPE` gültig:

**ls120** Boot von LS120/240 sowie ZIP Disks.

**hd** Boot von Festplatte. IDE und SATA Geräte werden direkt erkannt, für SCSI, USB oder besondere Controller wird das Paket `HD` und/oder `USB` benötigt. Näheres ist der Dokumentation (Seite ??) zum Paket `HD` zu entnehmen.

**cd** Boot von CD-ROM. Es wird lediglich das ISO-Image `fli4l.iso` der CD erzeugt, welches anschließend mit dem jeweiligen Lieblingsbrennprogramm selbst auf CD gebrannt werden muss. Bezüglich SCSI, USB und spezielle Controller ist das Paket `HD` bzw. `USB` nötig.

**integrated** Bei diesem Typ wird kein Bootmedium zu Grunde gelegt, sondern das OPT-Archiv vollständig ins RootFS integriert. Somit entfällt das Mounten des Bootmediums und der Kernel kann gleich alles entpacken. Dieser `BOOT_TYPE` wird z.B. fürs Booten vom Netzwerk benötigt.

**Hinweis:** Ein remote Update ist natürlich in diesem Fall nicht möglich.

**attached** Ähnlich wie **integrated**, jedoch wird das OPT-Archiv als Datei `opt.img` ans RootFS angehängt, mit der Folge, dass es wieder im Verzeichnis `/boot` zu finden ist und gesondert beim Bootvorgang entpackt wird. Ansonsten gilt das unter **integrated** Gesagte.

**netboot** Entspricht **integrated**. Es wird jedoch zusätzlich das Skript `mknetboot.sh` gestartet, welches ein Image zum Booten via LAN erzeugt. Weiteres ist bitte dem Wiki <https://ssl.networks.org/wiki/display/f/fli4l+und+Netzboot> zu entnehmen.



### 3. Basiskonfiguration

**pxeboot** Es werden zwei Images generiert, kernel und rootfs.img. Das sind die beiden vom PXE-Bootloader nachzuladenden Dateien. Beim Aufruf kann die Location des tftp-Verzeichnisses angegeben werden und zusätzlich noch ein Unterverzeichnis innerhalb des tftp-Verzeichnisses (-pxesubdir). Weiteres auch hier im Wiki <https://ssl.networks.org/wiki/display/f/fli4l+und+Netzboot>

**Hinweis:** wie ein fli4l als passender boot-server (pxe/tftp) zu konfigurieren ist, können sie in der Dokumentation des Pakets dns\_dhcp nachlesen.

**LIBATA\_DMA** Mit dieser Variable kann eingestellt werden, ob DMA für libata basierte Geräte aktiviert werden soll. Dies ist z.B. bei einigen unvollständig verdrahteten IDE zu Compact-Flash Adaptern nötig. Um DMA zu aktivieren: 'enabled' Default: 'disabled'

**MOUNT\_BOOT** Standardwert: MOUNT\_BOOT='rw'

Hier wird eingestellt, wie das Boot-Medium gemountet werden soll. Es gibt drei Möglichkeiten:

**rw** – Read/Write – Schreiben und Lesen ist möglich

**ro** – Read-Only – Nur Lesen ist möglich

**no** – None – Medium wird nach dem Boot wieder abgemeldet und kann dann bei Bedarf entnommen werden.

Bei bestimmten Konfigurationen ist es unbedingt erforderlich, das Medium Read/Write anzumelden, z.B. wenn man den DHCP-Server einsetzen oder die imond-Log-Datei auf dem Medium anlegen möchte.

**BOOTMENU\_TIME** Standardwert: BOOTMENU\_TIME='20'

Hier wird eingestellt, wie lange der syslinux Bootloader warten soll, bis automatisch mit der Standard-Installation gebootet wird.

Im Paket HD besteht die Möglichkeit, über OPT\_RECOVER eine Funktion zu aktivieren, mit der eine Notfallinstallation aus einer laufenden Installation erstellt werden kann. Diese kann im Bootmenü über die Wahl der Recover-Version aktiviert werden.

Sollte hier der Wert '0' eingestellt sein, wartet der syslinux Bootloader bis der Anwender die Standard- oder die Recover-Version auswählt und aktiviert!

**TIME\_INFO** Standardwert: TIME\_INFO='MEZ-1MESZ,M3.5.0,M10.5.0/3'

Uhren ticken in der Unix-Welt und damit auch unter fli4l normalerweise nach der UTC (Universal Time Coordinated), einer weltweit einheitlichen Uhrzeit, die vor der Verwendung in die lokale Zeit umgerechnet wird. TIME\_INFO liefert fli4l die dafür notwendigen Informationen über die Namen der Zeitzonen, die Differenz zu UTC und Regeln, wann auf Sommerzeit und wieder zurück gewechselt wird. Damit das korrekt funktioniert, muss die Hardware Uhr auf UTC gestellt werden (entspricht der Londoner Winterzeit) oder über das Paket chrony mit einem Timeserver synchronisiert werden (diese liefern UTC aus).

Die Einträge in TIME\_INFO bedeuten dabei folgendes:

### 3. Basiskonfiguration

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

- *MEZ-1*: Wir befinden uns in der mitteleuropäischen Zeitzone (*MEZ*), die der UTC eine Stunde voraus ist  $MEZ-1=UTC$ .
- *MESZ*: In dieser Zeitzone gibt es Sommerzeit (Mittleuropäische Sommerzeit). Da nichts weiter angegeben wird, kommt man zur Sommerzeit, indem man die Zeit eine Stunde vorstellt.
- *M3.5.0,M10.5.0/3*: Am letzten Sonntag im März (um 2 Uhr) wird zur Sommerzeit gewechselt, am letzten Sonntag im Oktober (um 3 Uhr) wieder zurück.

Normalerweise braucht man diesen Wert nie anzufassen, es sei denn man sitzt in einer anderen Zeitzone. Will man die Werte anpassen, sollte man einen Blick auf die Spezifikation der Umgebungsvariable TZ werfen, die unter folgender URL zu finden ist (englisch): [http://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd\\_chap08.html](http://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap08.html)

**RTC\_SYNC** Standardwert: `RTC_SYNC='hwclock'`

In vielen Rechnern steckt eine batteriegepufferte Hardware-Uhr, die auch über die Dauer der Abschaltung mit Strom versorgt wird und die Uhrzeit weiterzählt, so dass sie beim nächsten Starten wieder als Systemzeit zur Verfügung steht. Es ist an dieser Stelle wichtig, zwischen der *Systemzeit* und der *Hardwarezeit* zu unterscheiden:

- Die *Hardwarezeit* ist die Zeit, die in der Hardware-Uhr gespeichert und von dieser aktuell gehalten wird. Sie wird in der Regel beim Starten des Systems aus der Hardware-Uhr ausgelesen und als Systemzeit übernommen.
- Die *Systemzeit* ist die eigentliche Zeit, die das Linux-System verwendet und z. B. beim Aufruf des Befehls `date -u` angezeigt wird. Sie wird vom Linux-Kernel aktuell gehalten, etwa auf Basis regelmäßiger Hardware-Unterbrechungen (Timer-Interrupt), bezeichnet immer einen Zeitpunkt in koordinierter Weltzeit (UTC), und wird nicht von der Zeitzone-Einstellung beeinflusst.
- Die *lokalisierte Systemzeit* ist lediglich die Umrechnung der Systemzeit in eine andere Zeitzone, die auf dem fli4l-Router über die Umgebungsvariable TZ konfiguriert wird (siehe die Variable `TIME_INFO` (Seite 25)), und spielt im weiteren Verlauf dieses Abschnitts keine Rolle.

Mit Hilfe dieser Variable wird dem fli4l mitgeteilt, wie der Abgleich der Hardwarezeit mit der Systemzeit vorgenommen werden soll, d. h. ob und wie oft die Hardwarezeit auf die Systemzeit gesetzt werden soll. Ein solcher Abgleich ist nötig, weil auch die beste Hardware-Uhr nicht zu 100 % genau geht und zum systematischen Abdriften neigt, d. h. sie geht auf Dauer gesehen etwas zu langsam oder etwas zu schnell.

Es gibt prinzipiell zwei Möglichkeiten der Synchronisation:

- Modus “kernel”: Ein NTP-Client wird verwendet, um von außen (in der Regel über das Internet oder eine externe (Funk-)Uhr) die tatsächliche Uhrzeit zu ermitteln und die Systemzeit des fli4l-Routers aktuell zu halten. Dabei wird der Linux-Kernel angewiesen, sich um die Aktualisierung der Hardwarezeit zu kümmern, so dass keine weitere Synchronisierung mehr nötig ist. Die Aktualisierung durch den Linux-Kernel

### 3. Basiskonfiguration

ist etwas weniger genau als die Aktualisierung mittels **hwclock** (siehe Modus “hwclock” weiter unten), allerdings ist die Güte der Aktualisierung weit weniger wichtig, weil der zwangsläufige Fehler durch den NTP-Client ausgeglichen wird.

Dieser Modus muss auch verwendet werden, wenn gar keine Hardware-Uhr existiert. Der Linux-Kernel wird in diesem Falle natürlich keine Hardwarezeit aktuell halten, weil es keine gibt. Es sollte dann allerdings unbedingt ein NTP-Client verwendet werden, damit der fli4l-Router überhaupt eine sinnvolle Systemzeit erhält.

- Modus “hwclock”: Es findet beim Herunterfahren des Systems (bei der Ausführung des Stopp-Skripts `/etc/rc0.d/rc950.hwclock`) sowie in regelmäßigen Abständen (alle 24 Stunden) eine Synchronisation mit Hilfe des **hwclock**-Programms statt. Dabei wird nicht nur die Hardwarezeit gesetzt, sondern **hwclock** misst auch, inwieweit die Systemzeit von der Hardwarezeit abweicht. Beim Starten des Systems wird dann die Systemzeit nicht direkt aus der Hardwarezeit übernommen, sondern es wird auch die Abweichung berücksichtigt, um das Abdriften der Systemzeit möglichst zu reduzieren. Die Abweichung wird in der Datei `/etc/adjtime` vermerkt. Ist ein beschreibbares persistentes Medium verfügbar, wird die Abweichung unter `/var/lib/persistent/base/adjtime` gespeichert; in diesem Falle ist `/etc/adjtime` eine symbolische Verknüpfung dorthin.

Dieser Modus ist inkompatibel zu einer Aktualisierung der Systemzeit mit Hilfe eines NTP-Clients. Das liegt daran, dass ein NTP-Client automatisch das Aktualisieren der Hardwareuhr durch den Linux-Kernel aktiviert. Es ist jedoch wenig sinnvoll bzw. problematisch, dass sowohl **hwclock** als auch der Linux-Kernel gleichzeitig versuchen, die Hardwarezeit aktuell zu halten.

Es ist zu beachten, dass wenn eine Hardware-Uhr zur Verfügung steht, die darin gespeicherte Uhrzeit *immer* als koordinierte Weltzeit (UTC) interpretiert wird. Die Zeitzone, die über die Variable `TIME_INFO` gesetzt wird, wirkt sich nicht auf die in der Hardware-Uhr gespeicherte Zeit aus. Das Speichern einer lokalisierten Nicht-UTC-Zeit in der Hardware-Uhr wird von fli4l *nicht* unterstützt.

Das Ermitteln der Systemzeit aus der Hardwarezeit wird einmalig beim Starten des Systems vorgenommen. Dabei wird bereits durch den Linux-Kernel das Auslesen der Hardware-Uhr und das Setzen der Systemzeit unmittelbar am Beginn des Bootvorgangs vorgenommen. Im Modus “hwclock” wird dann später bei der Ausführung des Boot-Skripts `/etc/rc.d/rc100.hwclock` die Systemzeit erneut gesetzt, diesmal unter Berücksichtigung der systematischen Abweichung.

**KERNEL\_VERSION** Legt die Version des zu verwendenden Kerns fest. Entsprechend dieser Variable werden der Kern aus `img/kernel-<kernel version>.<compression extension>` und die Kernel-Module aus `opt/lib/modules/<kernel version>` selektiert.

**KERNEL\_BOOT\_OPTION** Standardwert: `KERNEL_BOOT_OPTION=""`

Der Inhalt dieser Variable wird an die Kommandozeile des Kerns in der `syslinux.cfg` angehängt. Manche Systeme benötigen für korrekten Reboot `'reboot=bios'`. Bei WRAP-Systemen also `'reboot=bios'`.

**COMP\_TYPE\_ROOTFS** Standardwert: `COMP_TYPE_ROOTFS='xz'`

### 3. Basiskonfiguration

Der Inhalt dieser Variable legt die Kompressionsmethode für das RootFS-Archiv fest. Mögliche Werte sind 'xz', 'lzma' und 'bzip2'.

**COMP\_TYPE\_OPT** Standardwert: `COMP_TYPE_OPT='xz'`

Der Inhalt dieser Variable legt die Kompressionsmethode für das OPT-Archiv fest. Mögliche Werte sind 'xz', 'lzma' und 'bzip2'.

**POWERMANAGEMENT** Standardwert: `POWERMANAGEMENT='acpi'`

Der Kern unterstützt verschiedene Formen des Powermanagements, das etwas betagte APM und das aktuellere ACPI. Hier kann man einstellen, welche Form er verwenden soll. Mögliche Werte sind 'none' (kein Powermanagement), 'acpi' und die beiden APM-Varianten 'apm' und 'apm\_rm'. Letzteres schaltet in einen speziellen Prozessormodus, bevor der Router ausgeschaltet wird.

**FLI4L\_UUID** Standardwert: `FLI4L_UUID=""`

Hier wird eine eindeutige UUID eingetragen, mit der der fli4l seine persistenten Daten auf z.B. einem USB-Stick finden kann. Eine UUID kann auf einem beliebigen Linux-System (wie auch dem fli4l) mit dem Befehl `'cat /proc/sys/kernel/random/uuid'` erstellt werden. Dies gibt bei jedem Aufruf eine neue UUID aus. Diese muss nun in die Variable eintragen werden. Auf einem persistenten Medium (z.B. auf einer Festplatte (OPT\_HD) oder einem USB-Stick (OPT\_USB und OPT\_HD)) muss dann noch ein Verzeichnis mit demselben Namen angelegt werden. Dort wird dann künftig alles gespeichert, das sich gegenüber der Konfiguration geändert hat, ebenso wie persistente Laufzeitdaten wie z.B. DHCP-Leases. Hierzu muss das entsprechende Paket dies natürlich unterstützen (siehe Dokumentation). Der entsprechende Eintrag für den Speicherpfad ist dort dann in der Regel 'auto'.

Sollte der fli4l bereits vor dem Erstellen der UUID und dem Anlegen des Verzeichnisses einige Daten gespeichert haben, so sind diese unter /boot/persistent zu finden und müssen dann manuell an den neuen Speicherort verschoben werden. Deshalb empfiehlt es sich, die UUID gleich anfangs zu erstellen und nicht erst später zu migrieren.

Zudem ist zu beachten, dass `MOUNT_BOOT='ro'` nicht gewählt werden darf, solange das Verzeichnis sich irgendwo auf der /boot Partition befindet.

Ein empfohlener Ort für das persistente Verzeichnis befindet sich auf der /data Partition (ganz oben) oder einem USB-Stick. Das Dateisystem des USB-Sticks sollte VFAT sein oder bei aktivem OPT\_HD alle dort unterstützen schreib-lese-fähigen Dateisysteme.

**IP\_CONNTRACK\_MAX** Standardwert: `IP_CONNTRACK_MAX=""`

Mit Hilfe dieser Variable kann man die maximale mögliche Anzahl gleichzeitiger Verbindungen manuell einstellen. Normalerweise wird anhand des eingebauten Arbeitsspeichers automatisch ein sinnvoller Wert ermittelt. In Tabelle 3.2 sind die verwendeten Voreinstellungen zusammengefasst dargestellt.

Bei Einsatz von FileSharing-Programmen hinter oder auf dem Router und wenig Arbeitsspeicher ist die maximale Anzahl gleichzeitiger Verbindungen aber sehr schnell erreicht und zusätzliche Verbindungen können nicht mehr aufgebaut werden.

Das äußert sich in Fehlermeldungen wie

### 3. Basiskonfiguration

Tabelle 3.2.: Automatische Einstellung der maximalen Verbindungsanzahl

Arbeitsspeicher in MiB	gleichzeitige Verbindungen
16	1024
24	1280
32	2048
64	4096
128	8192

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

oder

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

Mittels `IP_CONNTRACK_MAX` lässt sich nun die maximale Anzahl gleichzeitiger Verbindungen fest auf einen bestimmten Wert einstellen. Jede einzelne mögliche Verbindung kostet 350 Bytes Arbeitsspeicher, der nicht mehr für andere Dinge genutzt werden kann. Setzt man also 10000, so sind gerundet 3,34 MB Arbeitsspeicher für den normalen Gebrauch verloren (Kernel, Ramdisks, Programme).

Bei 32 MiB RAM sollte es kein Problem sein, mal eben 2 oder 3 MiB für die `ip_conntrack`-Tabelle zu reservieren, bei 16 MiB wird es knapp und bei 12 oder sogar 8 MiB ist absolute Sparwut angesagt.

Die momentan benutzte Einstellung lässt sich auf der Konsole mit

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

anzeigen und mit

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

zur Laufzeit setzen, wobei XXX für die Anzahl der Einträge steht. Die Einträge in der `IP_CONNTRACK`-Tabelle selbst können auf der Konsole mit

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

angesehen und mit

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

gezählt werden.

**LOCALE** Standardwert: `LOCALE='de'`

Einige Komponenten sind mittlerweile mehrsprachfähig. Dazu zählen beispielsweise das Konsolen-Menü und die Weboberfläche. Mit dieser Variablen können Sie die bevorzugte Sprache auswählen. Verschiedene Komponenten haben noch eine eigene Einstellung womit diese Grundeinstellung, wenn nötig, überlistet werden kann. Wenn die hier angegebene Sprache bei einer Komponente (noch) nicht verfügbar ist, wird auf Englisch zurückgefallen.

Bei `KEYBOARD_LOCALE='auto'` wird versucht ein zu der `LOCALE`-Einstellung passendes Tastatur-Layout ein zu stellen.

Bisher sind folgende Einstellungen möglich: de, en, es, fr, hu, nl.

### 3.3. Konsolen-Einstellungen

fli4l kann auf verschiedenen Hardware-Plattformen betrieben werden. Auf vielen dieser Plattformen ist es möglich, eine Tastatur und einen Monitor anzuschließen, um mit dem fli4l zu interagieren; diese Kombination der Ein- und Ausgabe wird generell *Konsole* genannt.

fli4l kann aber auch gänzlich ohne Tastatur und Grafikkarte eingesetzt werden. Damit man den Router dann auch ohne Netzwerkzugriff bedienen sowie alle Boot-Meldungen des Kernels sehen kann, ist es u. a. möglich, über die serielle Schnittstelle ebenfalls eine Konsole zu erhalten, indem die Ein- und Ausgaben von der serielle Schnittstelle bezogen bzw. dorthin ausgegeben werden. Dazu müssen die Variablen `SER_CONSOLE` (Seite 30), `SER_CONSOLE_IF` (Seite 31) und `SER_CONSOLE_RATE` (Seite 31) gesetzt bzw. angepasst werden.

Schließlich ist es möglich, zur selben Zeit sowohl über Tastatur und Monitor als auch über die serielle Schnittstelle eine Konsole zur Verfügung zu stellen.

Generell stellt fli4l auf *jeder* Konsole die Möglichkeit zur Anmeldung und somit eine *Shell* zur Verfügung, an der Sie sich mit dem Benutzer “fli4l” und dem über die Variable `PASSWORD` (Seite 24) konfigurierten Passwort anmelden können.

**CONSOLE\_BLANK\_TIME** Standard-Einstellung: `CONSOLE_BLANK_TIME=""`

Normalerweise aktiviert der Linux-Kern nach einer gewissen Zeit ohne Eingaben auf dem aktuellen Eingabegerät den Bildschirmschoner. Mit der Variable `CONSOLE_BLANK_TIME` kann man diese Zeit konfigurieren bzw. den Bildschirmschonermodus ganz deaktivieren (`CONSOLE_BLANK_TIME='0'`).

**BEEP** Standard-Einstellung: `BEEP='yes'`

Signalton am Ende des Boot- und Shutdownprozesses ausgeben.

Trägt man hier ‘yes’ ein, ertönt ein Signalton am Ende des Boot- und Shutdownprozesses. Wer extremen Platzmangel auf dem Bootmedium hat oder keinen Signalton möchte, kann hier also ‘no’ eingetragen lassen.

**SER\_CONSOLE** Standard-Einstellung: `SER_CONSOLE='no'`

Diese Variable aktiviert oder deaktiviert eine Konsole auf einer seriellen Schnittstelle. Die serielle Konsole kann in drei Modi betrieben werden:

SER_CONSOLE	Konsolen-Ein-/Ausgabe
no	Ein- und Ausgabe (nur) über Tastatur und Monitor (tty0)
yes	Ein- und Ausgabe (nur) über serielle Schnittstelle (ttyS0)
primary	Ein- und Ausgabe sowohl über serielle Konsole als auch über Tastatur und Monitor, Ausgabe der Kernelmeldungen auf tty0
secondary	Ein- und Ausgabe sowohl über serielle Konsole als auch über Tastatur und Monitor, Ausgabe der Kernelmeldungen auf ttyS0

Wenn der Wert von `SER_CONSOLE` verändert wird, wird diese Änderung nur bei der Erstellung eines neuen Bootmediums oder beim Remote-Update der `syslinux.cfg` wirksam.

**Wichtig:** Achten Sie beim Ausschalten der seriellen Konsole darauf, sich einen alternativen Zugang zum Router (SSH oder direkt über Tastatur und Monitor) zu erhalten!

Weitere Informationen sind im Anhang unter [Serielle Konsole](#) (Seite 141) zu finden.

**SER\_CONSOLE\_IF** Standard-Einstellung: `SER_CONSOLE_IF='0'`

Nummer der seriellen Schnittstelle für die serielle Konsole.

Hier ist die Nummer der Schnittstelle einzutragen, an die die serielle Konsole angeschlossen wird. 0 entspricht `ttyS0` unter Linux bzw. `COM1` unter Microsoft Windows.

**SER\_CONSOLE\_RATE** Standard-Einstellung: `SER_CONSOLE_RATE='9600'`

Übertragungsrate der seriellen Schnittstelle für die Konsolenausgabe.

Hier trägt man die Baud-Rate ein, mit der die Daten auf der seriellen Schnittstelle übertragen werden. Sinnvolle Werte: 4800, 9600, 19200, 38400, 57600, 115200.

## 3.4. Hilfen zum Einkreisen von Problemen und Fehlern

`fli4l` loggt die gesamten Ausgaben des Bootvorganges in einer Datei (`/var/tmp/boot.log`). Diese Datei kann man sich am Ende des Bootvorganges auf der Konsole oder über den entsprechenden Menüpunkt im Web-Interface ansehen.

Manchmal ist es jedoch sinnvoll, bei Problemen einen ausführlicheren Ablauf der Start-Sequenz zu generieren, um den Bootvorgang hinterher auf Probleme untersuchen zu können. Dazu dient `DEBUG_STARTUP`. Andere Einstellungen unterstützen Entwickler beim Finden von Fehlern in bestimmten Situationen; auch diese Einstellungen werden in diesem Abschnitt dokumentiert.

**DEBUG\_STARTUP** Standard-Einstellung: `DEBUG_STARTUP='no'`

Steht dieser Wert auf 'yes', wird beim Booten jedes ausgeführte Kommando vor seiner Ausführung auf den Schirm geschrieben. Da für das korrekte Funktionieren Änderungen an der `syslinux.cfg` vorgenommen werden müssen, gilt das für `SER_CONSOLE` Gesagte auch hier. Wenn man die `syslinux.cfg` von Hand ergänzen will, ist es nötig, ein `fli4ldebug=yes` einzufügen. `DEBUG_STARTUP` muss dann aber trotzdem auf 'yes' stehen.

**DEBUG\_MODULES** Standard-Einstellung: `DEBUG_MODULES='no'`

Einige Module werden automatisch vom Kern geladen, ohne dass man das vorher erkennen kann. `DEBUG_MODULES='yes'` aktiviert einen Modus, der einem die kompletten Modul-ladesequenzen zeigt, egal, ob sie von einem Skript oder vom Kern angestoßen werden.

**DEBUG\_ENABLE\_CORE** Standard-Einstellung: `DEBUG_ENABLE_CORE='no'`

Wird diese Option aktiviert, verursacht jeder Programmabsturz auf dem Router das Erzeugen einer so genannten “core”-Datei, also eines Speicherabbilds des Prozesses direkt vor dem Absturz. Diese Dateien sind auf dem Router im Verzeichnis `/var/log/dumps` zu finden. Diese Dateien können dann genutzt werden, um den Programmfehler besser zu finden. Genauer finden Sie hierzu im Abschnitt “Entwanzen von Programmen auf dem fli4l” (Seite ??) in der Dokumentation des SRC-Pakets.

**DEBUG\_MDEV** Standard-Einstellung: `DEBUG_MDEV='no'`

Mit `DEBUG_MDEV='yes'` werden alle Aktionen, die in Zusammenhang mit dem `mdev`-Dämon stehen und somit mit dem Hinzufügen oder Entfernen von Geräteknoten in `/dev` oder dem Laden von Firmware zu tun haben, in der Datei `/dev/mdev.log` protokolliert.

**DEBUG\_IPTABLES** Standard-Einstellung: `DEBUG_IPTABLES='no'`

Mit `DEBUG_IPTABLES='yes'` werden alle `iptables`-Aufrufe inklusive dem Rückgabewert in `/var/log/iptables.log` protokolliert.

**DEBUG\_IP** Standard-Einstellung: `DEBUG_IP='no'`

Diese Variable aktiviert bei `DEBUG_IP='yes'` das Protokollieren aller Aufrufe des Programms `/sbin/ip` in der Datei `/var/log/wrapper.log`.

## 3.5. Verwendung einer eigenen `/etc/inittab`

Man kann von Init zusätzliche Programme auf zusätzlichen Konsolen starten lassen oder die Standardkommandos der Init-Konfigurationsdatei verändern. Ein Eintrag sieht wie folgt aus:

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

Das *device* ist das Gerät, über das das Programm seine Ein-/Ausgaben machen soll. Möglich sind hier die normalen Terminals `tty1-tty4` oder serielle Terminals `ttyS0-ttySn` mit  $n < \text{Anzahl}$  der vorhandenen seriellen Schnittstellen.

Als *action* kommen in der Regel *askfirst* oder *respawn* in Frage. Bei *askfirst* wird auf einen Tastendruck gewartet, bevor das Kommando ausgeführt wird, bei *respawn* wird es automatisch neu gestartet, wenn sich das Programm beendet.

*command* ist das Programm, das ausgeführt werden soll. Es ist mit vollständiger Pfadangabe zu spezifizieren.

Die Busybox-Dokumentation unter <http://www.busybox.net> enthält eine genaue Beschreibung des `inittab` Formats.

Die normale `inittab` sieht wie folgt aus:



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

Diese könnte man z.B. um den Eintrag

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

erweitern, um ein zweites Login auf Terminal zwei zu bekommen. Dazu einfach die Datei `opt/etc/inittab` nehmen, nach `<configverzeichnis>/etc/inittab` kopieren und mit einem Editor bearbeiten.

## 3.6. Länderspezifische Tastaturlayouts

**KEYBOARD\_LOCALE** Standard-Einstellung: `KEYBOARD_LOCALE='auto'`

Wenn man öfter direkt auf dem fli4l Router arbeitet ist ein lokales Tastaturlayout eine willkommene Hilfe. Mit `KEYBOARD_LOCALE='auto'` wird versucht, ein Tastaturlayout passend zu der Einstellung von `LOCALE` zu benutzen. Mit der Einstellung `"` wird kein lokales Tastaturlayout auf dem fli4l-Router installiert; es wird dann das im Kernel anwesende Standardlayout verwendet. Alternativ kann man auch den Namen einer lokalen Tastaturlayoutmap direkt angeben. Wenn z.B. `'de-latin1'` eingestellt wird, prüft der Buildprozess ob in `opt/etc` eine Datei mit Namen `de-latin1.map` vorliegt. Wenn ja, wird die entsprechende `.map`-Datei als Tastaturlayout eingebunden.

**OPT\_MAKEKBL** Standard-Einstellung: `OPT_MAKEKBL='no'`

Wenn man für seine Tastatur eine map Datei erstellen will muss man wie folgt vorgehen:

- `OPT_MAKEKBL` auf `'yes'` setzen.
- Auf dem Router `'makekbl.sh'` anrufen. Vorzugsweise macht man dies über eine ssh-Verbindung weil das Tastaturlayout sich ändert und das lästig sein kann.
- Die Anweisungen befolgen.
- Ihre neue `<locale>.map` Datei liegt in der `/tmp`.
- Die Arbeiten direkt auf dem Router sind jetzt abgeschlossen.
- Nun kopieren Sie die eben erzeugte Tastaturtabelle in Ihr fli4l-Verzeichnis unter `opt/etc/<locale>.map`. Wenn Sie jetzt `KEYBOARD_LOCALE='<locale>'` setzen wird beim nächsten Buildprozess Ihre neu erzeugtes Tastaturlayout benutzt.
- Vergessen Sie nicht `OPT_MAKEKBL` wieder auf `'no'` zu setzen.

## 3.7. Ethernet-Netzwerkkarten-Treiber

**NET\_DRV\_N** Standard-Einstellung: `NET_DRV_N='1'`

### 3. Basiskonfiguration

Anzahl der benötigten Netzwerkkarten-Treiber.

Wird der Router nur für ISDN verwendet, so gibt es in der Regel nur eine Netzwerkkarte, der Standard-Wert ist also 1.

Bei DSL werden jedoch meistens zwei Netzwerkkarten verwendet.

Hierbei muss man zwei Fälle unterscheiden:

1. Beide Karten sind vom gleichen Typ. Dann muss nur ein Treiber geladen werden, der dann beide Karten anspricht, also `NET_DRV_N='1'`.
2. Es werden zwei verschiedene Karten verwendet, dann muss man hier '2' angeben und den Treiber für die zweite Karte angeben.

**NET\_DRV\_x** Standard-Einstellung: `NET_DRV_1='ne2k-pci'`

Hier wird der Treiber für die Netzwerkkarte angegeben. Über die Variable `NET_DRV_1` wird standardmäßig der Treiber für eine NE2000-kompatible Netzwerkkarte geladen. Weitere verfügbare Treiber für Netzwerkkartenfamilien sind in den Tabellen ?? und ?? eingetragen.

Die 3COM EtherLinkIII (3c509) muss über das Dostool `3c509cfg.exe` (beziehbar von <ftp://ftp.ihg.uni-duisburg.de/Hardware/3com/3C5x9n/3C5X9CFG.EXE>)

konfiguriert werden. Dabei sollten IRQ und I/O-Port und gegebenenfalls auch der Anschluss (BNC/TP) eingestellt werden.

**NET\_DRV\_x\_OPTION** Standard-Einstellung: `NET_DRV_x_OPTION=""`

Der Eintrag kann in der Regel leer bleiben.

Bei manchen ISA-Karten braucht der Treiber zusätzliche Informationen, um die Karte zu finden, z.B. die I/O-Adresse. Bei NE2000-kompatiblen ISA-Karten und bei der EtherExpress16 ist dies zum Beispiel der Fall.

Hier ist z.B.

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

zu setzen (oder der entsprechende numerische Wert).

Ist keine Option nötig, kann die Variable leer gelassen werden.

Sind mehrere Optionen nötig, sind diese durch Leerzeichen zu trennen, z.B.

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

Werden zwei identische Netzwerkkarten verwendet, z.B. NE2000-ISA-Karten, müssen die verschiedenen I/O-Werte durch Komma getrennt werden, also

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

### 3. Basiskonfiguration

Die beiden IO-Werte müssen durch Komma ohne Blank getrennt werden!

Dieses funktioniert nicht bei allen Netzwerkkarten-Treibern. Einige muss man auch doppelt laden, also dann `NET_DRV_N='2'`. In diesem Fall müssen aber mit der Option `“-o”` verschiedene Namen vergeben werden, z.B.

```
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'
```

Unser Tip: erst die Komma-Methode ausprobieren, danach das mehrfache Laden mit Option `“-o”` versuchen.

Nachfolgend noch einige Beispiele von Netzwerkkarten:

- 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=''
```

Siehe zu dieser Karte auch:

[http://extern.fli4l.de/fli4l\\_faengine/faq.php?display=faq&faqnr=132&catnr=7&prog=1](http://extern.fli4l.de/fli4l_faengine/faq.php?display=faq&faqnr=132&catnr=7&prog=1)

[http://extern.fli4l.de/fli4l\\_faengine/faq.php?display=faq&faqnr=133&catnr=7&prog=1](http://extern.fli4l.de/fli4l_faengine/faq.php?display=faq&faqnr=133&catnr=7&prog=1)

[http://extern.fli4l.de/fli4l\\_faengine/faq.php?display=faq&faqnr=135&catnr=7&prog=1](http://extern.fli4l.de/fli4l_faengine/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'
```

Oft muss man hier noch die IRQ-Werte mit angeben, also

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

Man sollte es hier zunächst ohne Angabe der Interrupts probieren und lediglich dann die Interrupts eintragen, wenn ohne Angabe der Interrupts die Netzwerkkarten nicht gefunden werden.

- 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'
```

Die vollständige Liste der verfügbaren Treiber finden Sie in der Dokumentation des jeweiligen Kernel-Pakets.

*Falls Sie ein dummy-Device brauchen, verwenden Sie 'dummy' für NET\_DRV\_x und IP\_NET\_x\_DEV (Seite 37)='dummy<Nummer>' als Device-Name.*

## 3.8. Netzwerk-Konfiguration (IPv4)

**OPT\_IPV4** Aktiviert die IPv4-Unterstützung.

**Wichtig:** Diese Einstellung muss zur Zeit den Wert 'yes' beinhalten! Ein reiner IPv6-Betrieb ist zur Zeit noch nicht möglich!

Standard-Einstellung: OPT\_IPV4='yes'

**IP\_NET\_N** Standardwert: IP\_NET\_N='1'

Anzahl der Netzwerke, die an das IPv4-Protokoll gebunden werden sollen. Im Normalfall also '1'. Falls es keine Netzwerke geben sollte oder sie über einen anderen Weg konfiguriert werden, und daher IP\_NET\_N auf 0 gesetzt wird, wird beim Bauen der Archive eine Warnung ausgegeben. Diese kann man durch setzen von IGNOREIPNETWARNING='yes' ausschalten.

**IP\_NET\_x** Standardwert: IP\_NET\_1='192.168.6.1/24'

Die IPv4-Adresse und die Netzmaske in CIDR<sup>1</sup>-Schreibweise des x'ten Devices im fli4l-Router. Wird die IPv4-Adresse dynamisch durch einen DHCP-Klienten zugewiesen, kann hier auch 'dhcp' als Wert eingetragen werden.

Nachfolgende Tabelle zeigt CIDR und Punkt-Schreibweise (DOT Notation).

---

<sup>1</sup>Classless Inter-Domain Routing

### 3. Basiskonfiguration

CIDR	Netzmaske	Anzahl IPs
/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

**Anmerkung:** Da jeweils eine IPv4 für die Netzwerk- und Broadcast-Adresse reserviert sind, errechnet sich die maximale Anzahl der Hosts im Netzwerk zu: `Anzahl_Hosts = Anzahl_IPs - 2`. Die kleinste mögliche Netzmaske wäre also /30, entspricht 4 IPs und somit 2 möglichen Hosts.

**IP\_NET\_x\_DEV** Standard-Einstellung: `IP_NET_1_DEV='eth0'`

Erforderlich: Device-Name der Netzwerkkarte.

Ab Version 2.1.8 ist die Angabe des verwendeten Device zwingend erforderlich! Die Namen der Devices beginnen in den meisten Fällen mit `'eth'` gefolgt von einer Zahl. Die erste vom System erkannte Netzwerkkarte bekommt den Namen `'eth0'`, die zweite `'eth1'` usw.

Beispiel:

```
IP_NET_1_DEV='eth0'
```

fl4l beherrscht auch IP-Aliasing, also die Zuweisung von mehreren IPs auf eine Netzwerkkarte. Zusätzliche IPs definiert man einfach mit einem weiteren Netzwerk auf dem selben Device. Beim Überprüfen der Konfiguration informiert mkfl4l darüber, dass man ein solches Alias definiert — diese Warnung kann man dann ignorieren.

Beispiel:

```
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** Standard-Einstellung: `IP_NET_1_MAC=""`

Optional: MAC-Adresse der Netzwerkkarte.

### 3. Basiskonfiguration

Hier kann die Hardwareadresse (MAC) der Netzwerkkarte angepasst werden. Dies ist zum Beispiel nützlich, wenn Sie einen DHCP-Provider nutzen wollen, der eine bestimmte MAC-Adresse erwartet. Wird `IP_NET_x_MAC` leer gelassen oder gar nicht angegeben, so wird die voreingestellte MAC-Adresse der Netzwerkkarte verwendet. Die allermeisten Nutzer werden diese Variable nicht benötigen.

Beispiel:

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

**IP\_NET\_x\_NAME** Standard-Einstellung: `IP_NET_x_NAME=""`

Optional: Der IPv4-Adresse der Netzwerkkarte einen Namen geben.

Bei umgekehrter Namensauflösung der IPv4-Adresse der Netzwerkkarte erscheint standardmässig ein Name der Form `'fli4l-ethx.<domain>'`. In `IP_NET_x_NAME` kann man selbst einen anderen Namen angeben. Dieser Name wird dann bei umgekehrter Namensauflösung angezeigt. Bei einer öffentlichen IPv4-Adresse kann man so erreichen, dass immer der öffentliche Name aufgelöst wird.

Beispiel:

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

**IP\_NET\_x\_TYPE**

**IP\_NET\_x\_COMMENT** Standard-Einstellung: `IP_NET_x_COMMENT=""`

Optional: Die Angabe dient dazu, einem Device einen 'aussagekräftigen' Namen zu geben. Dieser kann dann in Paketen wie z.B. `rrdtool` zur Identifikation des Netzwerks verwendet werden.

## 3.9. Netzwerk-Konfiguration (IPv6)

### 3.9.1. Einleitung

Zusätzlich zu der im letzten Abschnitt vorgestellten Konfiguration von IPv4-Netzwerken ist es auch möglich, den fli4l-Router in vielerlei Hinsicht IPv6-tauglich zu machen. Dazu gehören Angaben über die IPv6-Adresse des Routers, die verwalteten IPv6-(Sub-)Netze sowie vordefinierte IPv6-Routen und Firewall-Regeln bzgl. IPv6-Paketen. IPv6 ist der Nachfolger des Internet-Protokolls IPv4. Hauptsächlich wurde es entwickelt, um die relativ kleine Menge von eindeutigen Internet-Adressen zu vergrößern: Während IPv4 ungefähr  $2^{32}$  Adressen unterstützt,<sup>2</sup> sind es bei IPv6 bereits  $2^{128}$  Adressen. Dadurch kann mit IPv6 jedem kommunizierenden Host eine eindeutige Adresse zugeordnet werden, und man ist nicht mehr auf Techniken wie NAT, PAT, Masquerading etc. angewiesen.

---

<sup>2</sup>nur ungefähr, weil einige Adressen speziellen Zwecken dienen, etwa für Broad- und Multicasting

Neben diesem Aspekt spielten bei der Entwicklung des IPv6-Protokolls auch Themen wie Selbstkonfiguration und Sicherheit eine Rolle. Dies wird in späteren Abschnitten aufgegriffen.

Das größte Problem bei IPv6 ist dessen Verbreitung: Momentan wird IPv6 – verglichen mit IPv4 – nur sehr spärlich verwendet. Das liegt daran, dass IPv6 und IPv4 technisch nicht miteinander kompatibel sind und somit alle Software- und Hardware-Komponenten, die an der Paket-Weiterleitung im Internet beteiligt sind, für IPv6 nachgerüstet werden müssen. Auch bestimmte Dienste wie DNS (Domain Name System) müssen für IPv6 entsprechend aufgeböhrt werden.

Hier tut sich also ein Teufelskreis auf: Die geringe Verbreitung von IPv6 bei Diensteanbietern im Internet führt zu Desinteresse seitens der Router-Hersteller, ihre Geräte mit IPv6-Funktionalität auszustatten, was wiederum dazu führt, dass Dienstanbieter die Umstellung auf IPv6 scheuen, weil sie fürchten, dass sich der Aufwand nicht lohnt. Erst langsam wendet sich das Blatt zugunsten von IPv6, nicht zuletzt unter dem immer stärkeren Druck der knappen Adressvorräte.<sup>3</sup>

#### 3.9.2. Adressformat

Eine IPv6-Adresse besteht aus acht Zwei-Byte-Werten, die hexadezimal notiert werden:

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

*Beispiel 2:* 0:0:0:0:0:0:0:1 (IPv6-Loopback-Adresse)

Um die Adressen etwas übersichtlicher zu gestalten, werden aufeinander folgende Nullen zusammengelegt, indem sie entfernt werden und lediglich zwei unmittelbar aufeinander folgende Doppelpunkte verbleiben. Die obigen Adressen können also auch so geschrieben werden:

*Beispiel 1 (kompakt):* 2001:db8:900:551::2

*Beispiel 2 (kompakt):* ::1

Eine solche Kürzung ist aber nur höchstens einmal erlaubt, um Mehrdeutigkeiten zu vermeiden. Die Adresse 2001:0:0:1:2:0:0:3 kann also entweder zu 2001::1:2:0:0:3 oder zu 2001:0:0:1:2::3 verkürzt werden, nicht aber zu 2001::1:2::3, da jetzt unklar wäre, wie die vier Nullen jeweils auf die zusammengezogenen Bereiche verteilt werden sollen.

Eine weitere Mehrdeutigkeit existiert, wenn eine IPv6-Adresse mit einem Port (TCP oder UDP) kombiniert werden soll: In diesem Fall darf man den Port nicht unmittelbar mit Doppelpunkt und Wert anschließen, weil der Doppelpunkt bereits innerhalb der Adresse verwendet wird und somit in manchen Fällen unklar wäre, ob die Port-Angabe nicht vielleicht doch eine Adress-Komponente darstellt. Deshalb muss in solchen Fällen die IPv6-Adresse in eckigen Klammern angegeben werden. Dies ist auch die Syntax, wie sie in URLs gefordert wird (etwa wenn im Web-Browser eine numerische IPv6-Adresse verwendet werden soll).

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

Ohne die Verwendung von Klammern entsteht die Adresse 2001:db8:900:551::2:1234, die unverkürzt der Adresse 2001:db8:900:551:0:0:2:1234 entspricht und somit keine Port-Angabe besitzt.

Wenn Sie keinen Internet-Zugang mit nativer IPv6-Anbindung nutzen können, benötigen Sie einen IPv6-Tunnel zu einem IPv6-Anbieter. Dies wird durch das Paket “ipv6” ermöglicht. Bitte lesen Sie die Dokumentation des “ipv6”-Pakets für Details.

---

<sup>3</sup>Inzwischen sind die letzten IPv4-Adressblöcke von der IANA vergeben worden.

#### 3.9.3. Konfiguration

##### Allgemeine Einstellungen

Die allgemeinen Einstellungen beinhalten zum einen die Aktivierung der IPv6-Unterstützung und zum anderen die optionale Vergabe einer IPv6-Adresse an den Router.

**OPT\_IPV6** Aktiviert die IPv6-Unterstützung.

Standard-Einstellung: `OPT_IPV6='no'`

**HOSTNAME\_IP6** (optional) Diese Variable stellt explizit die IPv6-Adresse des Routers ein. Falls die Variable nicht gesetzt wird, wird die IPv6-Adresse auf die Adresse des ersten konfigurierten IPv6-Subnetzes gesetzt (`IPV6_NET_x`, siehe unten).

Beispiel: `HOSTNAME_IP6='IPV6_NET_1_IPADDR'`

##### Subnetz-Konfiguration

In diesem Abschnitt wird die Konfiguration von einem oder mehreren IPv6-Subnetzen vorgestellt. Ein IPv6-Subnetz ist ein IPv6-Adressraum, der über ein so genanntes Präfix spezifiziert wird und an eine bestimmte Netzwerk-Schnittstelle gebunden ist. Weitere Einstellungen betreffen das Veröffentlichen des Präfixes und des DNS-Dienstes innerhalb des Subnetzes sowie einen optionalen Router-Namen innerhalb dieses Subnetzes.

**IPV6\_NET\_N** Diese Variable enthält die Anzahl der verwendeten IPv6-Subnetze. Mindestens ein IPv6-Subnetz sollte definiert werden, um IPv6 im lokalen Netz nutzen zu können.

Standard-Einstellung: `IPV6_NET_N='0'`

**IPV6\_NET\_x** Diese Variable enthält für ein bestimmtes IPv6-Subnetz die IPv6-Adresse des Routers sowie die Größe der Netzmaske in CIDR-Notation. Soll das Subnetz öffentlich geroutet werden, so stammt es in der Regel vom Internet- bzw. Tunnel-Anbieter.

**Wichtig:** *Soll in dem Subnetz die zustandslose Selbstkonfiguration (siehe den Abschnitt zu `IPV6_NET_x_ADVERTISE` weiter unten) aktiviert werden, dann muss die Länge des Subnetz-Präfixes 64 Bit betragen!*

Ist das Subnetz an einen Tunnel angeschlossen oder wird DHCPv6 zum Ermitteln eines Präfixes verwendet, dann darf hier nur der Teil der Router-Adresse angegeben werden, der *nicht* zum dem Tunnel zugeordneten bzw. vom DHCPv6-Server erhaltenen Subnetz-Präfixes gehört, da jenes Präfix und diese Adresse miteinander kombiniert werden! Siehe die Dokumentation der “`ipv6`”- (Tunnel) bzw. “`dns_dhcp`”-Pakete (DHCPv6) für Details.

Beispiele:

<code>IPV6_NET_1='2001:db8:1743:42::1/64'</code>	# statisch: komplette Adresse
<code>IPV6_NET_2='{he}+0:0:0:42::1/64'</code>	# HE-Tunnel: partielle Adresse
<code>IPV6_NET_3='{dhcpv6}+0:0:0:43::1/64'</code>	# DHCPv6: partielle Adresse



**IPV6\_NET\_x\_DEV** Diese Variable enthält für ein bestimmtes IPv6-Subnetz den Namen der Netzwerk-Schnittstelle, an welche das IPv6-Netz gebunden wird. Dies kollidiert *nicht* mit den in der Basis-Konfiguration (`base.txt`) vergebenen Netzwerk-Schnittstellen, da einer Netzwerk-Schnittstelle sowohl IPv4- als auch IPv6-Adressen zugeordnet werden dürfen.

Beispiel: `IPV6_NET_1_DEV='eth0'`

**IPV6\_NET\_x\_ADVERTISE** Diese Variable legt fest, ob das eingestellte Subnetz-Präfix im LAN mittels “Router Advertisements” verbreitet wird. Dies wird für die so genannte “stateless autoconfiguration” (zustandslose Selbstkonfiguration) verwendet und ist nicht mit DHCPv6 zu verwechseln. Mögliche Werte sind “yes” und “no”.

Es ist empfehlenswert, diese Einstellung zu aktivieren, es sei denn, alle Adressen im Netz werden statisch vergeben oder ein anderer Router ist bereits dafür zuständig, das Subnetz-Präfix anzukündigen.

**Wichtig:** *Die automatische Verteilung des Subnetzes funktioniert nur, wenn das Subnetz ein /64-Netz ist, d.h. wenn die Länge des Subnetz-Präfixes 64 Bit beträgt! Der Grund hierfür ist, dass die anderen Hosts im Netzwerk ihre IPv6-Adresse aus dem Präfix und ihrer Host-MAC-Adresse berechnen und dies nicht funktioniert, wenn der Host-Anteil nicht 64 Bit beträgt. Wenn die Selbstkonfiguration fehlschlägt, sollte also geprüft werden, ob das Subnetz-Präfix nicht vielleicht falsch angegeben worden ist (z.B. als /48).*

Standard-Einstellung: `IPV6_NET_1_ADVERTISE='yes'`

**IPV6\_NET\_x\_ADVERTISE\_DNS** Diese Variable legt fest, ob auch der lokale DNS-Dienst im IPv6-Subnetz mittels “Router Advertisements” (RDNSS-Option) angekündigt werden soll. Dies funktioniert aber nur, wenn die IPv6-Funktionalität des DNS-Dienstes mit Hilfe der Variable `DNS_SUPPORT_IPV6='yes'` aktiviert ist. Mögliche Werte sind “yes” und “no”.

Standard-Einstellung: `IPV6_NET_1_ADVERTISE_DNS='no'`

**IPV6\_NET\_x\_NAME** (optional) Diese Variable legt einen Interface-spezifischen Hostnamen für den Router in diesem IPv6-Subnetz fest.

Beispiel: `IPV6_NET_1_NAME='fli4l-subnet1'`

## 3.10. Netzwerkpräfix-Konfiguration

**OPT\_NET\_PREFIX** Aktiviert die Unterstützung für selbstdefinierte Netzwerkpräfixe.

Ein Netzwerkpräfix ist technisch nichts anderes als die Adresse eines Netzwerks, allerdings steht es i. d. R. für ein Netz, das weiter unterteilt werden soll. Sinnvoll ist dies vor allem dann, wenn ein fli4l-Router ein Netzwerk nicht alleine verwaltet, sondern Teilnetze daraus anderen Routern zur Verwaltung überlässt. Durch die Definition und somit die Benennung des insgesamt zur Verfügung und Verteilung stehenden Netzes ist es möglich, die Netzwerk-Adresse an mehreren Stellen zu verwenden, ohne den gemeinsamen Präfix immer wieder hinschreiben zu müssen.

Konkrete Beispiele, wie man ein Netzwerkpräfix definiert, sind weiter unten bei den verschiedenen Typen von Netzwerkpräfixen zu finden.

Standard-Einstellung: `OPT_NET_PREFIX='yes'`

**NET\_PREFIX\_x** Über dieses Array werden die verschiedenen Netzwerkpräfixe definiert. Die einzelnen Komponenten werden im Anschluss erklärt.

**NET\_PREFIX\_x\_NAME** Name des Netzwerkpräfixes.

Diese Variable enthält den Namen des Präfixes. Dieser Name kann dann in Adressangaben verwendet werden, um das Präfix zu benutzen. Dabei wird der Name analog zu Circuit-Namen verwendet, d. h. er muss in geschweiften Klammern geschrieben werden.

**NET\_PREFIX\_x\_TYPE** Typ des Netzwerkpräfixes.

Diese Variable enthält den Typ des Präfixes. Die unterstützten Typen werden in Tab. 3.3 erläutert.

Typ	Bedeutung
static	Das Netzwerkpräfix wird direkt als feste Adresse angegeben.
generated-ula	Das Netzwerkpräfix ist eine vom fli4l generierte ULA <sup>4</sup> gemäß RFC 4193. <sup>5</sup> Wenn der fli4l Zugriff auf persistenten Speicher hat, dann wird das Präfix nur einmal generiert, so dass es auch über Neustarts des Routers hinweg stabil bleibt.

Tabelle 3.3.: Typen von Netzwerkpräfixen

#### 3.10.1. Netzwerkpräfixe vom Typ “stable”

Für Netzwerkpräfixe vom Typ “stable” gibt es die folgenden Einstellungen:

**NET\_PREFIX\_x\_STATIC\_IPV4 NET\_PREFIX\_x\_STATIC\_IPV6** Adresse(n) des Netzwerkpräfixes.

Mit Hilfe dieser Einstellung kann die IPv4- und/oder die IPv6-Adresse des Netzwerkpräfixes angegeben werden.

Beispiel:

```
NET {
  PREFIX {
    [] {
      NAME='site'
      TYPE='static'
      STATIC {
        IPV4='10.1.0.0/16'
        IPV6='fdce:1c35:301f::/48'
      }
    }
  }
}
```

<sup>4</sup>“Unique Local Address”

<sup>5</sup><https://tools.ietf.org/html/rfc4193>

### 3.10.2. Netzwerkpräfixe vom Typ “generated-ula”

Für Netzwerkpräfixe vom Typ “generated-ula” gibt es die folgenden Einstellungen:

**NET\_PREFIX\_x\_ULA\_DEV** Ethernet-Schnittstelle.

Mit Hilfe dieser Einstellung wird die Ethernet-Schnittstelle angegeben, deren MAC-Adresse für die Generierung der ULA herangezogen wird.

Beispiel:

```
NET {
  PREFIX {
    [] {
      NAME='site'
      TYPE='generated-ula'
      ULA {
        DEV='eth0'
      }
    }
  }
}
```

## 3.11. Zusätzliche Routen (IPv4)

**IP\_ROUTE\_N** Standard-Einstellung: `IP_ROUTE_N='0'`

Anzahl von zusätzlichen Netzwerkrouuten. Zusätzliche Netzwerkrouuten sind zum Beispiel dann erforderlich, wenn sich im LAN weitere Router befinden, über die andere Netzwerke erreichbar sein sollen.

Im Normalfall ist die Angabe von weiteren Netzwerkrouuten nicht erforderlich.

**IP\_ROUTE\_x** Die zusätzlichen Routen `IP_ROUTE_1`, `IP_ROUTE_2`, ... haben folgenden Aufbau:

```
network/netmaskbits gateway
```

Dabei ist `network` die Netzwerkadresse, `/netmaskbits` die Netzmaske in [CIDR](#) (Seite 37) Schreibweise und `gateway` die Adresse des Rechners, der die Verbindung zu dem Netzwerk herstellt. Der Gateway-Rechner muss natürlich im gleichen Netzwerk, wie der fli4l-Router liegen! Ist z.B. das Netzwerk 192.168.7.0 mit der Netzwerkmaste 255.255.255.0 über das Gateway 192.168.6.99 erreichbar, dann lautet der Eintrag:

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

Wenn der fli4l-Router nicht als Internet-Router eingesetzt wird sondern nur als reiner Ethernetrouter kann man in einem `IP_ROUTE_x` Eintrag eine Default-Route angeben. Dazu wird analog zu der `network/netmaskbits` Schreibweise `0.0.0.0/0` eingetragen, so wie im Beispiel zu sehen ist.

```
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'
```

Intern werden die hierüber spezifizierten Routen in Circuits (siehe [Circuits vom Typ “route”](#) (Seite 98)) umgewandelt. Dabei werden alle Routen mit demselben Gateway demselben Circuit zugeordnet.

## 3.12. Zusätzliche Routen (IPv6)

IPv6-Routen sind Wege für IPv6-Pakete. Damit der Router weiß, welches eingehende Paket er wohin schicken soll, greift er auf eine Routing-Tabelle zurück, in der genau diese Informationen zu finden sind. Im Falle von IPv6 ist es wichtig zu wissen, wohin IPv6-Pakete geschickt werden, die nicht ins lokale Netz sollen.

**IPV6\_ROUTE\_N** Diese Variable legt die Anzahl der zu spezifizierenden IPv6-Routen fest. In der Regel werden keine zusätzlichen IPv6-Routen benötigt.

Standard-Einstellung: `IPV6_ROUTE_N='0'`

**IPV6\_ROUTE\_x** Diese Variable enthält die Route in der Form ‘Zielnetz Gateway’, wobei das Zielnetz in der CIDR-Notation erwartet wird. Für die Default-Route muss als Zielnetz `::/0` verwendet werden.

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

## 3.13. Der Paketfilter (IPv4)

Der von fli4l verwendete Linux-Kern stellt einen Paketfilter zur Verfügung. Mit Hilfe dieses Paketfilters wird gesteuert, wer mit dem Router bzw. über ihn hinweg kommunizieren darf. Weiterhin können Dinge wie Port-Weiterleitung (ein an den Router gerichtetes Paket wird an einen anderen internen Rechner weitergereicht) und Maskierung (engl. “Masquerading”; Pakete, die von einem Rechner hinter dem Router kommen, werden so verändert, dass sie so aussehen, als kämen sie vom Router selbst) realisiert werden.

Die Struktur des Paketfilters ist in Abbildung 3.1 angedeutet. Pakete kommen über eine Netzwerk-Schnittstelle herein und durchlaufen die `PREROUTING`-Kette (eng. “chain”). Hier werden die an den Router gerichteten Pakete an einen anderen Rechner weitergereicht, indem die Zieladresse und der Zielport manipuliert werden. Ist das Paket an den Router gerichtet, wird es an die `INPUT`-Kette, andernfalls an die `FORWARD`-Kette weitergereicht. Beide Ketten prüfen, ob das Paket zulässig ist. Wird das Paket akzeptiert, wird es an den lokalen Zielprozess zugestellt oder über die `POSTROUTING`-Kette (in der das Maskieren von Paketen stattfindet) an diejenige Netzwerk-Schnittstelle weitergereicht, über die das Paket sein Ziel erreichen kann. Lokal generierte Pakete werden in der `OUTPUT`-Kette gefiltert und schließlich (falls erfolgreich) über die `POSTROUTING`-Kette ebenfalls an die korrekte Netzwerk-Schnittstelle weitergereicht.

Mit der Paketfilterkonfiguration lassen sich die einzelnen Ketten des Paketfilters direkt konfigurieren. Dazu gibt es für jede relevante Kette ein eigenes Array, d.h. eine für die `INPUT`-Kette (`PF_INPUT_%`), eine für die `FORWARD`-Kette (`PF_FORWARD_%`), eine für die `OUTPUT`-Kette

### 3. Basiskonfiguration



Abbildung 3.1.: Struktur des Paketfilters

(PF\_OUTPUT\_%), eine für die PREROUTING-Kette, in der die Portweiterleitung durchgeführt wird (PF\_PREROUTING\_%), und eine für die POSTROUTING-Kette, in der das Maskieren der Pakete durchgeführt wird (PF\_POSTROUTING\_%).

Ein Eintrag in einem dieser Arrays besteht im Wesentlichen aus einer Aktion (s.u.), die durch zusätzliche Bedingungen eingeschränkt werden kann. Diese Bedingungen beziehen sich auf Eigenschaften des betrachteten Paketes. Ein Paket enthält Informationen über seine Herkunft (Quelle, welcher Rechner hat das Paket losgeschickt), sein Ziel (an welchen Rechner und welche Anwendung soll das Paket gehen) u.a.m. Bedingungen können sich auf folgende Eigenschaften eines Paketes beziehen:

- die Quelle (Quell-Adresse, Quell-Port oder beides)
- das Ziel (Ziel-Adresse, Ziel-Port oder beides)
- das Protokoll
- die Schnittstelle, über welches das Paket hereinkommt bzw. hinausgeht
- die MAC-Adresse des Rechners, von dem das Paket kommt
- den Zustand des Paketes bzw. der Verbindung, zu der das Paket gehört

Kommt ein Paket herein, werden die Einträge bzw. die daraus generierten Regeln von oben nach unten abgearbeitet und die erste Aktion ausgeführt, bei der alle Bedingungen gelten. Trifft keine der Regeln zu, wird die Standardaktion ausgeführt, die man für (fast) jede Tabelle angeben kann.

Ein Eintrag hat dabei folgendes Format, wobei zu beachten ist, dass alle Einschränkungen optional sind:

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

An allen Stellen, an denen Netzwerke, IP-Adressen oder Hosts angegeben werden müssen, kann man sich auch auf IP\_NET\_%, IP\_NET\_%\_IPADDR oder via @hostname auf einen Host aus HOST\_% beziehen. Ist OPT\_DNS aktiviert, dann können außerhalb von Aktionen via @fqdn auch Hosts über ihren Namen referenziert werden, die *nicht* in HOST\_% zu finden sind. Das ist insbesondere dann sinnvoll, wenn es sich um externe Hosts handelt, die zudem viele (und wechselnde) IP-Adressen besitzen.

## 3.13.1. Aktionen des Paketfilters

Aktionen können die folgenden sein:

Aktion	Kette(n)	Bedeutung
ACCEPT	alle	Akzeptiere das Paket.
DROP	INPUT FORWARD OUTPUT	Verwirf das Paket (der Absender erkennt das nur daran, dass keine Antwort, aber auch keine Fehlermeldung zurückkommt).
REJECT	INPUT FORWARD OUTPUT	Weise das Paket zurück (der Absender erhält eine entsprechende Fehlermeldung).
LOG	alle	Protokolliere das Paket und gehe zur nächsten Regel. Um verschiedene Protokoll-Einträge auseinanderhalten zu können, kann ein Präfix verwendet werden, das via LOG:log-prefix angegeben wird. Dieses Präfix darf maximal 28 Zeichen lang sein und kann aus Buchstaben, Ziffern, dem Bindestrich (-) und dem Unterstrich (_) bestehen.
MASQUERADE	POSTROUTING	Maskiere das Paket: Ersetze die Quelladresse des Paketes durch die eigene, der Schnittstelle zugewiesenen Adresse und Sorge dafür, dass Antworten für diese Verbindung an den richtigen Rechner weitergeleitet werden.
SNAT	POSTROUTING	Ersetze die Quelladresse und den Quellport des Paketes durch die als Parameter für SNAT angegebene Adresse (für alle Pakete, die zu der gerade betrachteten Verbindung gehören).
DNAT	PREROUTING	Ersetze die Zieladresse und den Zielport des Paketes durch die als Parameter für DNAT angegebene Adresse (für alle Pakete, die zu der gerade betrachteten Verbindung gehören).
REDIRECT	PREROUTING OUTPUT	Ersetze den Zielport des Paketes durch den als Parameter für REDIRECT angegebenen Port und stelle das Paket lokal zu (für alle Pakete, die zu der gerade betrachteten Verbindung gehören).
NETMAP	PREROUTING POSTROUTING	Bilde die Ziel- bzw. Quelladresse des Paketes in den als Parameter für NETMAP angegebenen Bereich ab, die Ports bleiben unverändert (für alle Pakete, die zu der gerade betrachteten Verbindung gehören; in der PREROUTING-Kette wird die Zieladresse verändert, in der POSTROUTING-Kette die Quelladresse).

Tabelle 3.4.: Aktionen in Paketfilterregeln

Einige dieser Aktionen können durch die Optionen `BIDIRECTIONAL`, `LOG` oder `NOLOG` in ihrem Verhalten modifiziert werden. `BIDIRECTIONAL` generiert die gleiche Regel noch einmal, nur mit vertauschter Quell- und Zieladresse (und vertauschtem Quell- und Zielpport und/oder vertauschter ein- und ausgehender Netzwerk-Schnittstelle, falls angegeben). `LOG`/`NOLOG` aktivieren bzw. deaktivieren das Protokollieren für diese eine Regel.

### 3.13.2. Einschränkungen in den Regeln

Einschränkungen können durch die in den folgenden Abschnitten aufgeführten Bedingungen vorgenommen werden. Bei den Bedingungen kann man immer **any** angeben, wenn man an irgendeiner Stelle keine Einschränkung vornehmen will, aber trotzdem etwas angeben will/muss. Einschränkungen können in beliebiger Reihenfolge angegeben werden, wenn sie einen vorangestellten Präfix haben. Das gilt für alle Einschränkungen, außer für die Angabe einer Quell- bzw. Zieladresse. Diese müssen immer direkt vor der Aktion stehen, die anderen Einschränkungen müssen vorher erfolgen. Einschränkungen können auch negiert werden, dazu wird einfach ein **!** vorangestellt.

#### Einschränkungen der Quelle und des Ziels

Jedes Paket enthält eine Quell- und eine Zielangabe, jeweils in Form eines Tupels einer IP-Adresse und eines Ports.<sup>6</sup> Diese Quelle bzw. dieses Ziel kann für eine Einschränkung herangezogen werden. Die Angabe für die Quelle bzw. das Ziel kann folgendermaßen vorgenommen werden:

Ausdruck	Bedeutung
<code>ip</code>	eine einfache IP-Adresse
<code>network</code>	eine Netzwerkangabe der Form <code>&lt;ip&gt;/&lt;netmask&gt;</code>
<code>port[-port]</code>	ein Port bzw. ein Port-Bereich
<code>IP_NET_x_IPADDR</code>	die IP-Adresse der Schnittstelle <code>x</code> des Routers
<code>IP_NET_x</code>	das Subnetz <code>x</code> des Routers
<code>IP_ROUTE_x</code>	das in der Route <code>x</code> angegebene Subnetz (Default-Routen können nicht verwendet werden, sie würden <b>any</b> entsprechen und werden vorsichtshalber ausgeklammert)
<code>@name</code>	einer der via <code>HOST_%*</code> vergebenen Namen oder Aliase; es wird die zugehörige IP-Adresse an dieser Stelle eingesetzt
<code>&lt;ip oder network&gt;:port[-port]</code>	Host- bzw. Netzwerk-Adresse in einer der obigen Varianten, kombiniert mit einem Port bzw. Port-Bereich

Tabelle 3.5.: Quell- und Zieleinschränkungen in Paketfilterregeln

<sup>6</sup>Ein Port ist nur bei TCP- und UDP-Paketen vorhanden.

### 3. Basiskonfiguration

Das könnte z. B. wie folgt aussehen: `'192.168.6.2 any DROP'`

Tauchen zwei dieser Angaben auf, wird die erste als Quelle, die zweite als Ziel betrachtet. In diesem Beispiel verwerfen wir also Pakete, die vom Rechner mit der IP-Adresse 192.168.6.2 gesendet wurden, unabhängig davon, an welches Ziel sie gerichtet sind.

Taucht nur eine Angabe auf, wird anhand des Wertes entschieden, ob die Quelle oder das Ziel gemeint ist, wobei die Entscheidung relativ einfach ist:

- Ist eine Port-Angabe enthalten, ist das Ziel gemeint.
- Sonst ist die Quelle gemeint.

Wenn wir also z. B. das obige Beispiel abkürzen wollten, könnten wir einfach `'192.168.6.2 DROP'` schreiben. Es ist kein Port angegeben, die Bedingung gilt also für die Quelle, den Rechner, von dem das Paket gesendet wurde.

Wollen wir die Kommunikation mit dem `ssh`-Dämon erlauben, können wir `'any any:22 ACCEPT'` (Pakete von einem beliebigen Rechner an den `ssh`-Port 22 eines beliebigen Rechners werden akzeptiert) oder kürzer `'22 ACCEPT'` schreiben: Es ist nur ein Port angegeben, also meinen wir das Ziel und damit Pakete, die an den Port 22 gerichtet sind.

Zur Vereinfachung der Regelmenge kann man an die Aktion ein `BIDIRECTIONAL` dranhängen, um auszudrücken, dass die Regel für beide Kommunikationsrichtungen gilt. Es werden dann Regeln generiert, in denen einfach die Quell- und die Ziel-Adressen und eventuell angegebenen Ports und Netzwerk-Schnittstellen vertauscht sind und der Rest gleich bleibt.

Beispiele:

<code>127.0.0.1 ACCEPT</code>	Lokale Kommunikation (Quelle 127.0.0.1) ist erlaubt
<code>any 192.168.12.1 DROP</code>	Pakete an die Adresse 192.168.12.1 werden weggeworfen
<code>any 192.168.12.1 DROP LOG</code>	Pakete an die Adresse 192.168.12.1 werden weggeworfen und zusätzlich protokolliert
<code>any 192.168.12.1 DROP NOLOG</code>	Pakete an die Adresse 192.168.12.1 werden weggeworfen, werden aber nicht protokolliert
<code>22 ACCEPT</code>	Pakete an den Port 22 ( <code>ssh</code> ) werden akzeptiert
<code>IP_NET_1_NET ACCEPT</code>	Pakete aus dem an der ersten Schnittstelle hängenden Subnetz werden akzeptiert
<code>IP_NET_1_NET IP_NET_2_NET ACCEPT BIDIRECTIONAL</code>	Kommunikation zwischen den an der ersten und zweiten Schnittstelle hängenden Subnetzen ist gestattet

#### Einschränkung der Schnittstelle

Eine Regel kann eingeschränkt werden in Bezug auf die Schnittstelle, über die ein Paket herkam bzw. hinausgeht. Das Format sieht wie folgt aus: `if:in:out`

In der `INPUT`-Kette kann man die Schnittstelle für hinausgehende Pakete nicht einschränken (das Paket geht ja nicht mehr hinaus), in der `POSTROUTING`-Kette kann man die Schnittstelle für hereinkommende Pakete nicht einschränken, da die Information darüber nicht mehr vorhanden ist. Lediglich in der `FORWARD`-Kette kann man für beides Bedingungen angeben.

Möglich sind folgende Werte für *in* bzw. *out*:

- `lo` (Loopback-Schnittstelle, lokale Kommunikation auf dem Router)
- `IP_NET_x_DEV`



- `pppoe` (die PPPoE-Schnittstelle; nur bei entsprechend aktiviertem `dsl`- oder `pppoe_server`-Paket)
- `any`

#### Einschränkungen des Protokolls

Eine Regel kann eingeschränkt werden in Bezug auf das Protokoll, zu dem ein Paket gehört. Das Format sieht wie folgt aus: `prot:protocol` bzw. `prot:icmp:icmp-type`. `protocol` kann dabei einen der folgenden Werte annehmen:

- `tcp`
- `udp`
- `gre` (Generic Routing Encapsulation)
- `icmp` (hier kann man zusätzlich noch einen Namen für den zu filternden ICMP-Typ angeben (`echo-reply` oder `echo-request`), etwa `prot:icmp:echo-request`)
- numerischer Wert der Protokoll-ID (z. B. 41 für IPv6)
- `any`

Wenn eine solche Einschränkung nicht vorhanden ist, aber dennoch Portnummern in einer Regel verwendet werden, dann wird die Regel *zweimal* angelegt, nämlich einmal für das `tcp`- und einmal für das `udp`-Protokoll.

#### Einschränkung der MAC-Adresse

Mittels `mac:mac-address` kann eine Einschränkung bezüglich der MAC-Adresse vorgenommen werden.

#### Einschränkungen in Bezug auf den Zustand eines Paketes

Der von `fi4l` verwendete Paketfilter sammelt Informationen über den Zustand von Verbindungen. Diese Informationen kann man dann nutzen, um Pakete zu filtern, also z. B. nur Pakete durchzulassen, die zu bereits bestehenden Verbindungen gehören. Die Zustände einer Verbindung können sein:<sup>7</sup>

Zustand	Bedeutung
INVALID	Das Paket gehört zu keiner bekannten Verbindung.
ESTABLISHED	Das Paket gehört zu einer Verbindung, über die bereits in beide Richtungen Pakete geflossen sind.
NEW	Das Paket hat eine neue Verbindung aufgebaut oder gehört zu einer Verbindung, bei der noch nicht Pakete in beide Richtungen geflossen sind.

<sup>7</sup>siehe [http://www.sns.ias.edu/~jns/files/iptables\\_talk/x38.htm](http://www.sns.ias.edu/~jns/files/iptables_talk/x38.htm) für eine genauere Beschreibung der Zustände

Zustand	Bedeutung
RELATED	Das Paket baut eine neue Verbindung auf, hat aber eine Beziehung zu einer bestehenden Verbindung (z.B. baut <code>ftp</code> eine separate Verbindung für den eigentlichen Datentransfer auf).

Tabelle 3.6.: Zustandseinschränkungen in Paketfilterregeln

Die Zustände werden wie folgt angegeben: `state:state(s)`. Will man mehrere Zustände angeben, werden diese mit Komma getrennt. Um z. B. nur Pakete durchzulassen, die direkt oder indirekt zu bestehenden Verbindungen gehören, kann man `state:ESTABLISHED,RELATED` schreiben (dies ist in der `INPUT`- oder `FORWARD`-Kette sinnvoll).

### Einschränkung der Häufigkeit einer Aktion

Unter bestimmten Umständen möchte man die Häufigkeit von Aktionen begrenzen, z. B. nur eine ICMP-Echo-Anforderung pro Sekunde zulassen. Das kann man mit der `limit`-Einschränkung erreichen, die wie folgt aussieht: `limit:Häufigkeit:Burst`. Die Häufigkeit wird dabei als  $n/\text{Zeiteinheit}$  (second, minute, hour, day) angegeben, wobei zusätzlich noch Ereignisse gehäuft auftreten können (Burst). Die Angabe `limit:3/minute:5` heißt z. B., dass höchstens drei Ereignisse pro Minute erlaubt sind, wobei auch mal fünf Ereignisse dicht aufeinander folgend akzeptiert werden.

### 3.13.3. Der Einsatz von Schablonen im Paketfilter

Um den Umgang mit dem Paketfilter weiter zu vereinfachen, gibt es die Möglichkeit, häufig vorkommende Regeln zu sogenannten Schablonen (Templates) zusammenzufassen. Damit ist es möglich, eine ganze Reihe von Paketfilterregeln zusammenzufassen und dieser Sammlung von Regeln einen symbolischen Namen zu geben. Anstatt direkt mit Protokollen und Portnummern zu hantieren, verwenden Sie dann Einträge wie `tmpl:ssh`, wenn Sie das `ssh`-Protokoll in einer Regel verwenden wollen. Wie mit Schablonen zu verfahren ist, wird hier am Beispiel von `ssh` gezeigt.

Wollen Sie Ihren `fi4l`-Router vom Internet aus per `ssh` erreichen, so schreiben Sie in einen Eintrag der Array-Variable `PF_INPUT_%` den entsprechenden Dienstenamen (hier `ssh`) mit vorangestelltem `tmpl:` und der Aktion, die für diesen Dienst gelten soll. Beispiel:

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

Hierbei steht `tmpl:` dafür, dass eine Regel auf einer Schablone basieren soll. Den Namen des Dienstes geben Sie nach dem `:` an, in unserem Beispiel also `ssh`. Zum Schluss geben Sie an, welche Aktion mit dem Dienst verbunden werden soll. Da Sie den `fi4l`-Router aus dem Internet erreichen wollen, erlauben wir die Verbindung mit `ACCEPT`. Einschränkungen von IP-Adressen oder Netzen sind nicht angegeben, also ist der `ssh`-Dienst von allen Netzen und über alle Schnittstellen erreichbar. Sie könnten bei Bedarf die gewohnten Schreibweisen vom Paketfilter benutzen, um den Zugriff auf den `ssh`-Dienst weiter einzuschränken.

Für welche Dienste Regeln vorbereitet sind (d.h. Schablonen existieren), kann in der Schablonen-Datei in `opt/etc/fwrules.tmpl/templates` nachgelesen werden. Im Folgenden finden Sie die Aufstellung in Tabellenform (siehe Tabelle 3.7).

### 3. Basiskonfiguration

Schablone	Protokoll	Port(s)
ad	tcp	389
ad	udp	389
ad	tcp	636
ad	tcp	3268
ad	tcp	3269
ad	udp	88
ad	tcp	88
ad	udp	53
ad	tcp	53
ad	udp	445
ad	tcp	445
ad	tcp	135
ad	tcp	5722
ad	udp	123
ad	udp	464
ad	tcp	464
ad	udp	138
ad	tcp	9389
ad	udp	67
ad	udp	2535
ad	udp	137
ad	udp	139
checkmk	tcp	6556
checkmk	tcp	161
checkmk	udp	161
checkmk	tcp	162
checkmk	udp	162
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

### 3. Basiskonfiguration

Schablone	Protokoll	Port(s)
ipmi	udp	623
irc	tcp	6667
ldap	tcp/udp	389
mail	tcp	110
mail	tcp	143
mail	tcp	25
mail	tcp	465
mail	tcp	587
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

### 3. Basiskonfiguration

Schablone	Protokoll	Port(s)
xmppserver	tcp	5269

Tabelle 3.7.: Im Lieferumfang von fli4l enthaltene Schablonen

Die Syntax für diese Form der Paketfilterregeln lautet also immer

```
tmpl:<Name des Dienstes> <Einschränkungen> <Gewünschte Aktion>
```

wobei als **<Einschränkungen>** alles erlaubt ist, was unter [3.13.2](#) beschrieben wird. Die möglichen Werte für **<Gewünschte Aktion>** sind in [3.13.1](#) aufgelistet und beschrieben.

Ein paar weitere Beispiele sollen die Arbeitsweise verdeutlichen. Zuerst wollen wir uns PF\_PREROUTING ansehen:

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

Die Regel PF\_PREROUTING\_1 versorgt die Xbox mit allem, was für Xbox Live notwendig ist. Im einzelnen werden mit `tmpl:xbl` alle Ports und Protokolle, die für Xbox Live notwendig sind, an den Host `xbox` weitergeleitet. Anstelle der IP-Adresse wird ein Eintrag aus dem `HOST_%NAME-`Array benutzt. Durch `dynamic` weiß der fli4l, dass die Ports von der Internet-Schnittstelle weitergeleitet werden sollen.

Die zweite Regel leitet das `https`-Protokoll an einen Webserver in der DMZ weiter.

Jetzt sehen wir uns an, wie es mit PF\_INPUT weitergeht:

```
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'
```

Die erste Regel lässt alle aus dem Netz, das mit `IP_NET_1` definiert ist, auf den Router zugreifen. Die zweite Regel ist für das `oident`-Paket. Dort wird der `ident`-Port geöffnet. Die dritte und letzte Regel erlaubt der Xbox den Zugriff auf den DNS Server auf dem fli4l. Hier ist auch wieder schön zu sehen, wie man einen Host-Alias einsetzt.

In PF\_FORWARD und PF\_POSTROUTING steht nichts `tmpl`-spezifisches mehr.

Es ist auch möglich, dass Sie eigene Schablonen anlegen oder dass Pakete ihre eigenen Schablonen mitbringen. Um eine eigene Schablone anzulegen, muss lediglich eine Datei mit den Namen der Schablone erstellt und die entsprechenden Regeln dort aufgenommen werden. Wenn Sie eine private Schablonendatei anlegen wollen, erstellen Sie diese in dem Verzeichnis `etc/fwrules.tmpl` unterhalb Ihres `config`-Verzeichnisses, so wie es die Abbildung [3.2](#) zeigt. Wenn das Verzeichnis `etc/fwrules.tmpl` unterhalb Ihres `config`-Verzeichnisses noch nicht existiert, legen Sie bitte zuerst beide Verzeichnisse an. Alternativ können Paket-Entwickler oder Benutzer, die Schablonen für mehr als eine Konfiguration anlegen wollen, ihre Regeln direkt in das Verzeichnis `opt/etc/fwrules.tmpl` ablegen. In dieses Verzeichnis kommen dann die neuen Schablonen. Dabei gilt die Regel, dass die Schablonen im `config`-Verzeichnis des Benutzers vorrangig behandelt werden. Zum Schluss wird die Schablonendatei, die zum Lieferumfang von fli4l gehört, ausgewertet. Sie können also Einträge in der fli4l-Schablonendatei



Abbildung 3.2.: Verzeichnisstruktur fli4l

dadurch „überschreiben“, indem Sie eine Schablonendatei mit dem Namen der zu überschreibenden Schablone in Ihrem `config`-Verzeichnis anlegen.

Wenn Sie zum Beispiel die Schablone `vpn_freunde` anlegen wollen, legen Sie die Datei `vpn_freunde` an. Das Template soll die Dienste `ssh`, `smtp`, `dns` und `samba` enthalten. Also schreiben Sie in die Datei `vpn_freunde` Folgendes:

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

Wann immer Sie jetzt die Schablone `vpn_freunde` benutzen, werden daraus Regeln für alle darin aufgeführten Protokolle und Ports erzeugt. `PF_FORWARD_x='tmpl:vpn_freunde ACCEPT'` etwa erstellt folgende `FORWARD`-Regeln:

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

#### 3.13.4. Die Konfiguration des Paketfilters

Der Paketfilter wird im Wesentlichen durch vier Array-Variablen konfiguriert:

### 3. Basiskonfiguration

- `PF_INPUT_%` konfiguriert die `INPUT`-Kette,
- `PF_FORWARD_%` konfiguriert die `FORWARD`-Kette,
- `PF_OUTPUT_%` konfiguriert die `OUTPUT`-Kette,
- `PF_PREROUTING_%` konfiguriert die `PREROUTING`-Kette und
- `PF_POSTROUTING_%` konfiguriert die `POSTROUTING`-Kette.

Für alle folgenden Ketten gilt die in `PF_LOG_LEVEL` vorgenommene Einstellung der Protokoll-Stufe, deren Inhalt auf einen der folgenden Werte gesetzt werden kann: `debug`, `info`, `notice`, `warning`, `err`, `crit`, `alert`, `emerg`.

#### Die `INPUT`-Kette

Über die `INPUT`-Kette wird konfiguriert, wer auf den Router zugreifen darf. Trifft keine der Regeln der `INPUT`-Kette zu, bestimmt die Standard-Aktion, was mit dem Paket passieren soll, und die Protokoll-Variable bestimmt, ob es bei einer Ablehnung ins System-Protokoll geschrieben werden soll.

Bei den verwendeten Parametern gibt es die folgenden Einschränkungen:

- Es können nur `ACCEPT`, `DROP` und `REJECT` als Aktion angegeben werden.
- Bei einer Schnittstellen-Einschränkung kann man nur die Eingangsschnittstelle einschränken.

**`PF_INPUT_POLICY`** Diese Variable beschreibt die Standard-Aktion, die angewandt wird, wenn keine der anderen Regeln zutrifft. Möglich sind:

- `ACCEPT` (nicht empfohlen)
- `REJECT`
- `DROP` (nicht empfohlen)

**`PF_INPUT_ACCEPT_DEF`** Steht diese Variable auf 'yes', werden Standard-Regeln generiert, die für ein korrektes Funktionieren des Routers notwendig sind. Standardmäßig sollte man hier 'yes' eintragen.

Möchte man das Verhalten komplett selbst definieren, kann man hier 'no' eintragen, muss dann jedoch alle Regeln selbst definieren. Eine zum Standardverhalten äquivalente Konfiguration würde wie folgt aussehen (die Beschreibung der Liste für benutzerdefinierte Ketten erfolgt [hier](#) (Seite 59)):

```
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'
```

### 3. Basiskonfiguration

```
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'
```

Die erste Regel verzweigt zur ratenlimitierenden “usr-in-icmp”-Kette. Die zweite Regel akzeptiert nur solche Pakete, die zu bestehenden Verbindungen gehören (also Paketen, die entweder den Zustand **ESTABLISHED** oder **RELATED** besitzen), und die dritte erlaubt lokale Kommunikation (**if:lo:any ACCEPT**). Die vierte filtert Pakete heraus, die behaupten, lokale Kommunikation zu sein, aber nicht bereits von der vorherigen Regel akzeptiert wurden.

Arbeitet man mit OpenVPN, muss man die Regeln noch ergänzen, um die von diesen Paketen verwendeten Ketten einzubinden.

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

**PF\_INPUT\_LOG** Definiert, ob abgelehnte Pakete vom Kernel protokolliert werden sollen. Dabei können die Meldungen durch Aktivierung von **OPT\_KLOGD** über den syslog-Dämon entsprechend der Konfiguration ausgegeben werden.

**PF\_INPUT\_LOG\_LIMIT** Definiert, wie häufig Log-Einträge generiert werden. Die Häufigkeit wird analog zur Limit-Einschränkung als  $n/\text{Zeiteinheit}$  mit Bursts beschrieben, also z. B. **3/minute:5**. Ist dieser Eintrag leer, wird der Standardwert **1/second:5** verwendet; enthält er **none**, wird keine Limitierung durchgeführt.

**PF\_INPUT\_REJ\_LIMIT PF\_INPUT\_UDP\_REJ\_LIMIT** Definiert, wie häufig bei einer Ablehnung eines hereinkommenden Paketes auch ein entsprechendes **REJECT**-Paket generiert wird. Die Häufigkeit wird analog zur Limit-Einschränkung als  $n/\text{Zeiteinheit}$  mit Bursts beschrieben, also z. B. **3/minute:5**. Ist das Limit überschritten, wird das Paket einfach ignoriert (**DROP**). Ist dieser Eintrag leer, wird der Standardwert **1/second:5** verwendet; enthält er **none**, wird keine Limitierung durchgeführt.

**PF\_INPUT\_ICMP\_ECHO\_REQ\_LIMIT** Definiert, wie häufig auf eine ICMP-Echo-Anfrage reagiert werden soll. Die Häufigkeit wird analog zur Limit-Einschränkung als  $n/\text{Zeiteinheit}$  mit Bursts beschrieben, also z. B. **/minute:5**. Ist das Limit überschritten, wird das Paket einfach ignoriert (**DROP**). Ist dieser Eintrag leer, wird der Standardwert **1/second:5** verwendet; enthält er **none**, wird keine Limitierung durchgeführt.

**PF\_INPUT\_ICMP\_ECHO\_REQ\_SIZE** Definiert, wie groß eine empfangene ICMP-Echo-Anfrage sein darf (in Bytes). In dieser Angabe sind neben den “Nutzdaten” auch die Paket-Header mit zu berücksichtigen. Der Standard-Wert liegt bei 150 Bytes.

**PF\_INPUT\_N PF\_INPUT\_x PF\_INPUT\_x\_COMMENT** Liste der Regeln, die beschreiben, welche Pakete vom Router angenommen bzw. verworfen werden.



#### Die FORWARD-Kette

Über die FORWARD-Kette wird konfiguriert, welche Pakete vom Router weitergeleitet werden. Trifft keine der Regeln der FORWARD-Kette zu, bestimmt die Standard-Aktion, was mit dem Paket passieren soll, und die Protokoll-Variable bestimmt, ob es bei einer Ablehnung ins System-Protokoll geschrieben werden soll.

Bei den verwendeten Parametern gibt es die Einschränkung, dass nur ACCEPT, DROP und REJECT als Aktion angegeben werden können.

**PF\_FORWARD\_POLICY** Diese Variable beschreibt die Standard-Aktion, die angewandt wird, wenn keine der anderen Regeln zutrifft. Möglich sind:

- ACCEPT
- REJECT
- DROP

**PF\_FORWARD\_ACCEPT\_DEF** Bestimmt, ob der Router Pakete akzeptiert, die zu bestehenden Verbindungen gehören. Steht diese Variable auf 'yes', generiert fli4l automatisch eine Regel, die Pakete mit dem entsprechenden Zustand akzeptiert:

```
'state:ESTABLISHED,RELATED ACCEPT',
```

weiterhin eine Regel, die Pakete mit unbekanntem Zustand verwirft:

```
'state:INVALID DROP'.
```

und schließlich eine Regel, die Pakete mit gefälschten IP-Adressen verwirft:

```
'state:NEW 127.0.0.1 DROP BIDIRECTIONAL'.
```

Zusätzlich generieren die anderen Subsysteme auch noch Standardregeln – eine Konfiguration ohne Standardregeln mit Portweiterleitung und OpenVPN würde mindestens folgende Regeln enthalten:

```
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** Definiert, ob abgelehnte Pakete vom Kernel protokolliert werden sollen. Dabei können die Meldungen durch Aktivierung von OPT\_KLOGD über den syslog-Dämon entsprechend der Konfiguration ausgegeben werden.

**PF\_FORWARD\_LOG\_LIMIT** Definiert, wie häufig Log-Einträge generiert werden. Die Häufigkeit wird analog zur Limit-Einschränkung als  $n/\text{Zeiteinheit}$  mit Bursts beschrieben, also z. B. 3/minute:5. Ist dieser Eintrag leer, wird der Standardwert 1/second:5 verwendet; enthält er none, wird keine Limitierung durchgeführt.

**PF\_FORWARD\_REJ\_LIMIT PF\_FORWARD\_UDP\_REJ\_LIMIT** Definiert, wie häufig bei einer Ablehnung eines hereinkommenden Paketes auch ein entsprechendes REJECT-Paket generiert wird. Die Häufigkeit wird analog zur Limit-Einschränkung als  $n/\text{Zeiteinheit}$  mit Bursts beschrieben, also z. B. `3/minute:5`. Ist das Limit überschritten, wird das Paket einfach ignoriert (DROP). Ist dieser Eintrag leer, wird der Standardwert `1/second:5` verwendet; enthält er `none`, wird keine Limitierung durchgeführt.

**PF\_FORWARD\_N PF\_FORWARD\_x PF\_FORWARD\_x\_COMMENT** Liste der Regeln, die beschreiben, welche Pakete vom Router weitergeleitet bzw. verworfen werden.

#### Die OUTPUT-Kette

Über die OUTPUT-Kette wird konfiguriert, worauf der Router selbst zugreifen darf. Trifft keine der Regeln der OUTPUT-Kette zu, bestimmt die Standard-Aktion, was mit dem Paket passieren soll, und die Protokoll-Variable bestimmt, ob es bei einer Ablehnung ins System-Protokoll geschrieben werden soll.

Bei den verwendeten Parametern gibt es die folgenden Einschränkungen:

- Es können nur ACCEPT, DROP und REJECT als Aktion angegeben werden.
- Bei einer Schnittstellen-Einschränkung kann man nur die Ausgangsschnittstelle einschränken.

**PF\_OUTPUT\_POLICY** Diese Variable beschreibt die Standard-Aktion, die angewandt wird, wenn keine der anderen Regeln zutrifft. Möglich sind:

- ACCEPT
- REJECT
- DROP

**PF\_OUTPUT\_ACCEPT\_DEF** Steht diese Variable auf 'yes', werden Standard-Regeln generiert, die für ein korrektes Funktionieren des Routers notwendig sind. Standardmäßig sollte man hier 'yes' eintragen.

Möchte man das Verhalten komplett selbst definieren, kann man hier 'no' eintragen, muss dann jedoch alle Regeln selbst definieren. Eine zum Standardverhalten äquivalente Konfiguration würde wie folgt aussehen:

```
PF_OUTPUT_ACCEPT_DEF='no'

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

Die erste (und einzige) Regel akzeptiert nur solche Pakete, die zu bestehenden Verbindungen gehören (also Paketen, die entweder den Zustand ESTABLISHED oder RELATED besitzen).

**PF\_OUTPUT\_LOG** Definiert, ob abgelehnte Pakete vom Kernel protokolliert werden sollen. Dabei können die Meldungen durch Aktivierung von OPT\_KLOGD über den syslog-Dämon entsprechend der Konfiguration ausgegeben werden.

**PF\_OUTPUT\_LOG\_LIMIT** Definiert, wie häufig Log-Einträge generiert werden. Die Häufigkeit wird analog zur Limit-Einschränkung als  $n/\text{Zeiteinheit}$  mit Bursts beschrieben, also z.B. `3/minute:5`. Ist dieser Eintrag leer, wird der Standardwert `1/second:5` verwendet; enthält er `none`, wird keine Limitierung durchgeführt.

**PF\_OUTPUT\_REJ\_LIMIT PF\_OUTPUT\_UDP\_REJ\_LIMIT** Definiert, wie häufig bei einer Ablehnung eines hereinkommenden Paketes auch ein entsprechendes REJECT-Paket generiert wird. Die Häufigkeit wird analog zur Limit-Einschränkung als  $n/\text{Zeiteinheit}$  mit Bursts beschrieben, also z.B. `3/minute:5`. Ist das Limit überschritten, wird das Paket einfach ignoriert (DROP). Ist dieser Eintrag leer, wird der Standardwert `1/second:5` verwendet; enthält er `none`, wird keine Limitierung durchgeführt.

**PF\_OUTPUT\_N PF\_OUTPUT\_x PF\_OUTPUT\_x\_COMMENT** Liste der Regeln, die beschreiben, welche Pakete vom Router versandt bzw. verworfen werden.

#### Benutzerdefinierte Listen

Aus verschiedenen Gründen besteht manchmal der Bedarf, eigene Ketten anzulegen und dort die Pakete genauer zu filtern. Diese Ketten kann man mittels `PF_USR_CHAIN_%` definieren und mit Regeln füllen. Die Namen der Ketten müssen dabei mit *usr-* beginnen und können nach ihrer Definition überall in der INPUT- oder FORWARD-Kette statt einer Aktion eingesetzt werden. Als Beispiel soll hier die bereits vorher verwendete ICMP-Filterkette dienen:

```
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** Definiert die Anzahl der benutzerdefinierten Ketten.

**PF\_USR\_CHAIN\_x\_NAME** Definiert den Namen der benutzerdefinierten Kette. Dieser muss mit *usr-* beginnen.

**PF\_USR\_CHAIN\_x\_RULE\_N**

**PF\_USR\_CHAIN\_x\_RULE\_x**

**PF\_USR\_CHAIN\_x\_RULE\_x\_COMMENT** Hier werden die Regeln definiert, die in die benutzerdefinierte Kette eingefügt werden sollen. Es können alle Regeln verwendet werden, die auch in einer FORWARD-Kette verwendet werden könnten. Sollte keine Regel der benutzerdefinierten Kette zutreffen, wird zur Ausgangskette zurückgekehrt und mit der Regel nach der Verzweigung fortgefahren.

#### Die NAT-Ketten (Network Address Translation)

Pakete können vor und nach Routing-Entscheidungen noch manipuliert werden. Sie können zum Beispiel eine neue Zieladresse erhalten, um an einen anderen Rechner weitergeleitet zu werden (Portweiterleitung) oder eine andere Quelladresse erhalten, um das hinter dem Router liegende Netzwerk zu maskieren. Maskieren nutzt man beispielsweise, um ein privates Netz über eine öffentliche IP ins Netz zu bringen oder in einem DMZ-Setup die Struktur des lokalen Netzes vor den Rechnern in der DMZ zu verbergen.

Die Konfiguration erfolgt über zwei Ketten, die `PREROUTING`- und die `POSTROUTING`-Kette. Über die `POSTROUTING`-Kette wird konfiguriert, welche Pakete vom Router maskiert werden. Trifft keine der Regeln der `POSTROUTING`-Kette zu, werden die Pakete unmaskiert weitergeleitet.

Beim Maskieren gibt es zwei Varianten: eine für Netzwerk-Schnittstellen, die bei der Auswahl erst eine IP-Adresse zugewiesen bekommen (`MASQUERADE`) und eine für Netzwerk-Schnittstellen mit statischer IP-Adresse (`SNAT`). `SNAT` erwartet dabei zusätzlich die IP-Adresse, die im Paket als Quelle eingetragen werden soll. Diese kann als

- IP-Adresse (Beispiel: `SNAT:1.2.3.4`),
- IP-Bereich (Beispiel: `SNAT:1.2.3.4-1.2.3.10`)
- oder als symbolische Referenz (Beispiel: `SNAT:IP_NET_1_IPADDR`)

angegeben werden.

Sowohl bei `SNAT` als auch bei `MASQUERADE` kann schließlich ein Port bzw. Portbereich angegeben werden, auf den der Quellport abgebildet werden soll. Normalerweise ist das nicht nötig, da der Kern die Ports allein auswählen kann. Es gibt aber Anwendungen, die verlangen, dass der Quellport unverändert bleibt (und somit ein 1:1-NAT erfordern) oder die ein PAT (Port Address Translation) oder NAT (Network Address and Port Translation) verbieten. Der Portbereich wird einfach hinten angehängt, z. B. so: `SNAT:IP_NET_1_IPADDR:4000-8000`.

Bei der `POSTROUTING`-Kette können nur `ACCEPT`, `SNAT`, `NETMAP` und `MASQUERADE` als Aktionen verwendet werden.

#### `PF_POSTROUTING_N PF_POSTROUTING_x PF_POSTROUTING_x_COMMENT`

Eine Liste der Regeln, die beschreiben, welche Pakete vom Router maskiert werden (bzw. unmaskiert weitergeleitet werden). Will man Pakete vom Maskieren ausklammern, kann man eine `ACCEPT`-Regel für die auszuklammernden Pakete der `MASQUERADE`-Regel voranstellen.

Über die `PREROUTING`-Kette wird konfiguriert, welche Pakete an einen anderen Rechner weitergeleitet werden sollen. Trifft keine der Regeln der `PREROUTING`-Kette zu, werden die Pakete unverändert weiterbehandelt. Die Aktion `DNAT` erwartet dabei die IP-Adresse, die im Paket als Ziel eingetragen werden soll. Diese kann als

- IP-Adresse (Beispiel: `DNAT:1.2.3.4`),
- IP-Bereich (Beispiel: `DNAT:1.2.3.4-1.2.3.10`)
- oder als Hostname (Beispiel: `DNAT:@client1`)

angegeben werden.

Schließlich kann noch ein Port bzw. Portbereich angegeben werden, auf den der Zielport abgebildet werden soll. Das ist aber nur nötig, wenn der Port geändert werden soll. Der Port bzw. Portbereich wird einfach hinten angehängt, z. B. so: DNAT:@server:21.

REDIRECT verhält sich wie DNAT, nur dass die Ziel-IP-Adresse immer auf die (primäre) IP-Adresse der Schnittstelle, auf der das Paket hereinkam, gesetzt wird und damit das Paket lokal zugestellt wird. Dies wird z. B. für transparente Proxys benötigt, siehe OPT\_TRANSPROXY (Seite ??).

Will man eine Portweiterleitung auf Schnittstellen mit dynamischen Adressen machen, weiß man zum Zeitpunkt der Konfiguration noch nicht, an welche IP die Pakete gerichtet sein werden. Daher kann man in der PREROUTING-Kette **dynamic** als Platzhalter für die später zugewiesene IP verwenden, etwa wie folgt:

```
'dynamic:80 DNAT:1.2.3.4'          # leite http-Pakete an die
                                   # IP-Adresse 1.2.3.4 weiter
'prot:gre any dynamic DNAT:1.2.3.4' # leite gre-Pakete (Teil des PPTP-
                                   # Protokolls) an die IP-Adresse
                                   # 1.2.3.4 weiter
```

Bei der PREROUTING-Kette können nur ACCEPT, DNAT, NETMAP und REDIRECT als Aktionen verwendet werden.

Für weitere Beispiele zur Portweiterleitung siehe den nächsten Abschnitt.

#### **PF\_PREROUTING\_N PF\_PREROUTING\_x PF\_PREROUTING\_x\_COMMENT**

Eine Liste der Regeln, die beschreiben, welche Pakete vom Router an ein anderes Ziel weitergeleitet werden sollen.

#### **3.13.5. Beispiele**

Im Folgenden sind einige Beispiele für die Paketfilter-Konfiguration angegeben.

##### **Die fli4l-Standardkonfiguration**

Die fli4l-Standardkonfiguration der Distribution sieht für die INPUT-Kette wie folgt aus:

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

Damit erreichen wir, dass

- Rechner im lokalen Netzwerk auf den Router zugreifen dürfen (PF\_INPUT\_1='IP\_NET\_1 ACCEPT'),
- lokale Kommunikation auf dem Router erlaubt ist (PF\_INPUT\_ACCEPT\_DEF='yes'),
- Pakete, die zu vom Router aufgebauten Verbindungen gehören, akzeptiert werden (PF\_INPUT\_ACCEPT\_DEF='yes'),

### 3. Basiskonfiguration

- alles andere abgelehnt wird (PF\_INPUT\_POLICY='REJECT'),
- aber nicht ins System-Protokoll geschrieben wird (PF\_INPUT\_LOG='no').

Für die FORWARD-Kette sieht das so ähnlich aus: Nur Pakete unseres lokalen Netzes und Pakete, die zu Verbindungen gehören, die von Rechnern im lokalen Netz aufgebaut wurden, sollen weitergeleitet werden. Des Weiteren werden NetBIOS- und CIFS-Pakete verworfen.

```
PF_FORWARD_POLICY='REJECT'  
PF_FORWARD_ACCEPT_DEF='yes'  
PF_FORWARD_LOG='no'  
PF_FORWARD_N='2'  
PF_FORWARD_1='tmp1:samba DROP'  
PF_FORWARD_2='IP_NET_1 ACCEPT'
```

Was man hier gut sieht, ist die Abhängigkeit von der Reihenfolge der Regeln: *Zuerst* werden NetBIOS-Pakete verworfen, und *danach* werden die Pakete des lokalen Netzes akzeptiert.

Nun kann das lokale Netz mit dem Router kommunizieren, seine Pakete werden weitergeleitet, es fehlt nur noch das Maskieren, welches für den Zugriff eines privaten Netzwerkes auf das Internet notwendig ist:

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

#### Trusted Nets

Wollen wir lokal mehrere Subnetze haben, die frei und unmaskiert miteinander kommunizieren können, müssen wir dafür sorgen, dass Pakete zwischen diesen Subnetzen nicht verworfen und auch nicht maskiert werden. Dazu fügen wir einfach eine Regel hinzu oder modifizieren die vorhandene.

Angenommen, wir haben einen DSL-Zugang über PPPoE, und die beiden Subnetze sind IP\_NET\_1 (192.168.6.0/24) und IP\_NET\_2 (192.168.7.0/24). Dann würde die Konfiguration wie folgt aussehen:

```
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='tmp1: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'
```

Regel eins sorgt jetzt dafür, dass Pakete zwischen den beiden Subnetzen ohne weitere Prüfung weitergeleitet werden. Die Regeln drei und vier sorgen dafür, dass beide Subnetze auch ins Internet kommen. Die erste Regel der POSTROUTING-Kette sorgt dafür, dass die Kommunikation zwischen den Subnetzen unmaskiert erfolgt.

### 3. Basiskonfiguration

Alternativ könnten wir auch sagen, dass nur Pakete, die über die **pppoe**-Schnittstelle hinausgehen, maskiert werden sollen:

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

Genauso hätte man die Filterung der Ports auch auf die **pppoe**-Schnittstelle beschränken und die beiden Subnetze zu einem zusammenfassen können, das würde dann wie folgt aussehen:

```
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'
```

Pakete, die über die **pppoe**-Schnittstelle hinausgehen und die an die **udp**-Ports 137-138 oder an die **tcp**-Ports 139 und 445 adressiert sind, werden verworfen (Regel 1), alle anderen Pakete, die aus dem Subnetz 192.168.6.0/23 kommen, werden weitergeleitet (Regel 2).

#### Route Network

Fügen wir dem Ganzen noch ein Netzwerk 10.0.0.0/24 hinzu (z. B. ein Dial-In-Netzwerk), mit dem wir unmaskiert kommunizieren wollen, wobei Pakete an die **udp**-Ports 137-138 sowie an die **tcp**-Ports 139 und 445 verworfen werden sollen, dann würde das wie folgt aussehen:

```
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'
```

- Regel 1 erlaubt die ungehinderte Kommunikation zwischen den Subnetzen **IP\_NET\_1** und **IP\_NET\_2**.
- Regel 2 verwirft Pakete an die Samba-Ports.
- Die Regeln 3 und 4 erlaubt die Weiterleitung von Paketen, die aus den Subnetzen 192.168.6.0/24, 192.168.7.0/24 und 10.0.0.0/24 kommen; die Rückrichtung wird von der Einstellung **PF\_FORWARD\_ACCEPT\_DEF='yes'** abgedeckt.
- Regel 1 der **POSTROUTING**-Kette sorgt dafür, dass Pakete in das bzw. aus dem 10.0.0.0/24-Subnetz nicht maskiert werden.

### 3. Basiskonfiguration

Alternativ ginge auch:

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

Diese Regel besagt, dass nur Pakete, die über die **pppoe**-Schnittstelle hinausgehen, maskiert werden.

#### Blacklists, Whitelists

Blacklists (ein Rechner in dieser Liste darf etwas nicht) und Whitelists (ein Rechner in dieser Liste darf etwas) werden prinzipiell ähnlich umgesetzt. Es werden Regeln geschrieben, die am Anfang sehr speziell sind und nach hinten immer allgemeiner werden. Bei einer Blacklist stehen am Anfang Regeln, die etwas verbieten und am Ende Regeln, die allen bisher nicht erwähnten etwas erlauben. Bei einer Whitelist ist es genau umgekehrt.

*Beispiel 1:* Alle Rechner im Subnetz 192.168.6.0/24 außer Rechner 12 dürfen ins Internet, solange sie nicht mit den CIFS Ports 137-138 (**udp**), 139 und 445 (**tcp**) kommunizieren wollen:

```
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'
```

*Beispiel 2:* Nur Rechner 12 darf ins Internet (aber nicht an die o.g. Ports ...), alle anderen dürfen nur lokal mit einem anderen Subnetz kommunizieren:

```
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'
```

#### 3.13.6. Standardkonfigurationen

##### Einfacher maskierender Router mit einem Netz dahinter

```
#  
# Zugriff auf den Router  
#  
PF_INPUT_POLICY='REJECT'  
PF_INPUT_ACCEPT_DEF='yes'
```



### 3. Basiskonfiguration

```
PF_INPUT_LOG='no'
PF_INPUT_N='1'
PF_INPUT_1='IP_NET_1 ACCEPT'    # alle Hosts im lokalen Netz dürfen
                                # auf den Router zugreifen

#
# Zugriff auf das ``Internet''
#
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'

PF_FORWARD_N='2'
PF_FORWARD_1='tmpl:samba DROP' # Samba-Pakete, die das Netz
                                # verlassen wollen, werden verworfen
PF_FORWARD_2='IP_NET_1 ACCEPT' # alle anderen Pakete dürfen das
                                # lokale Netz verlassen

#
# Maskieren des lokalen Netzes
#
PF_POSTROUTING_N='1'
PF_POSTROUTING_1='IP_NET_1 MASQUERADE' # maskiere Pakete, die das Subnetz
                                         # verlassen
```

#### Einfacher maskierender Router mit zwei Netzen dahinter

```
#
# Zugriff auf den 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'    # alle Hosts im lokalen Netz dürfen
                                # auf den Router zugreifen
PF_INPUT_2='IP_NET_2 ACCEPT'    # alle Hosts im lokalen Netz dürfen
                                # auf den Router zugreifen

#
# Zugriff auf das ``Internet''
#
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'

#
# Freie Kommunikation zwischen den Netzen
#
PF_FORWARD_N='4'
PF_FORWARD_1='IP_NET_1 IP_NET_2 ACCEPT BIDIRECTIONAL'
PF_FORWARD_2='tmpl:samba DROP' # Samba-Pakete, die das Netz
                                # verlassen wollen, werden verworfen
```

### 3. Basiskonfiguration

```
PF_FORWARD_3='IP_NET_1 ACCEPT' # alle anderen Pakete dürfen das
                                # lokale Netz verlassen
PF_FORWARD_4='IP_NET_2 ACCEPT' # alle anderen Pakete dürfen das
                                # lokale Netz verlassen

#
# Maskieren der lokalen Netze, unmaskierte Kommunikation zwischen den
# Netzen
#
PF_POSTROUTING_N='3'
PF_POSTROUTING_1='IP_NET_1 IP_NET_2 ACCEPT BIDIRECTIONAL'
PF_POSTROUTING_2='IP_NET_1 MASQUERADE' # maskiere Pakete, die das Subnetz
                                         # verlassen
PF_POSTROUTING_3='IP_NET_2 MASQUERADE' # maskiere Pakete, die das Subnetz
                                         # verlassen
```

#### **Maskierender DSL-Router mit zwei Netzen dahinter und SSH/HTTP-Zugriff aus dem Internet**

```
#
# Zugriff auf den 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' # alle Hosts im lokalen Netz dürfen
                              # auf den Router zugreifen
PF_INPUT_2='IP_NET_2 ACCEPT' # alle Hosts im lokalen Netz dürfen
                              # auf den Router zugreifen
PF_INPUT_3='tmpl:ssh ACCEPT' # gestatte Zugriff auf SSH-Dienst
                              # von überall her
PF_INPUT_4='tmpl:http 1.2.3.4/24 ACCEPT' # gestatte Rechner aus
                              # einem bestimmten Subnetz Zugriff
                              # auf HTTP-Dienst

#
# Zugriff auf das ``Internet''
#
PF_FORWARD_POLICY='REJECT'
PF_FORWARD_ACCEPT_DEF='yes'
PF_FORWARD_LOG='no'

#
# Keine Kommunikation zwischen den Netzen, beide Netze dürfen ins
# Internet, Samba-Pakete werden verworfen
#
PF_FORWARD_N='2'
PF_FORWARD_1='tmpl:samba if:any:pppoe DROP' # Samba-Pakete, die das Netz
                                              # verlassen wollen, werden verworfen
PF_FORWARD_2='if:any:pppoe ACCEPT' # alle anderen Pakete dürfen das
                                    # lokale Netz verlassen
```

### 3. Basiskonfiguration

```
#
# Maskieren der lokalen Netze, unmaskierte Kommunikation zwischen den
# Netzen
#
PF_POSTROUTING_N='1'
PF_POSTROUTING_1='if:any:pppoe MASQUERADE' # maskiere Pakete, die das Subnetz
                                           # verlassen
```

#### Portweiterleitung

Portweiterleitungen lassen sich mit den PREROUTING-Regeln wie folgt umsetzen (TARGET bezeichnet die ursprüngliche Zieladresse (optional) und den ursprünglichen Zielpport, NEW\_TARGET bezeichnet die neue Zieladresse und den neuen Zielpport (optional), PROTOCOL bezeichnet das jeweilige Protokoll):

```
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>'
```

#### Transparenter Proxy

Will man bestimmte Zugriffe auf das Internet nur über einen lokalen Proxy zulassen, kann man das mit Hilfe der PREROUTING- und POSTROUTING-Ketten erzwingen, ohne dass der Client davon etwas merkt. Prinzipiell sind dazu drei Schritte notwendig:

1. Anfragen an den HTTP-Port, die nicht vom Proxy kommen, an den Proxy umleiten (PREROUTING).
2. Die umgeleiteten Pakete so verändern, dass der Proxy denkt, sie kommen vom Router, so dass er sie wieder dorthin zurückschickt (POSTROUTING).
3. Die Pakete durch die FORWARD-Kette durchlassen, sofern ein Eintrag à la

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

nicht existiert (FORWARD).

*Beispiel 1:* Angenommen, wir haben nur ein Netz IP\_NET\_1, in dem auf einem Rechner namens proxy ein Squid-Proxy läuft, und wollen den gesamten http-Datenverkehr über ihn

### 3. Basiskonfiguration

leiten. Squid lauscht auf Port 3128. Der Einfachheit halber beziehen wir uns via `@proxy` auf den eingetragenen Host aus `HOST_1_NAME='proxy'` (vgl. [Domainkonfiguration](#) (Seite 77)).

Das Ganze würde wie folgt aussehen:

```
...
PF_PREROUTING_x='@proxy ACCEPT'
    # Pakete vom Proxy sollen nicht umgeleitet werden

PF_PREROUTING_x='prot:tcp IP_NET_1 80 DNAT:@proxy:3128'
    # HTTP-Pakete aus IP_NET_1 mit einem beliebigen Ziel werden
    # umgeleitet nach @proxy, Port 3128

PF_POSTROUTING_x='any @proxy:3128 SNAT:IP_NET_1_IPADDR'
    # alle Pakete an den Proxy-Port 3128 so umschreiben, als wären sie
    # vom fli4l (IP_NET_1_IPADDR)

PF_FORWARD_x='prot:tcp @proxy 80 ACCEPT'
    # HTTP-Pakete vom Proxy durch die FORWARD-Kette durchlassen (wenn nötig)
...
```

Gibt es mehrere Netze oder potentielle Konflikte mit anderen Portweiterleitungen (die ja auch nichts anderes sind als DNAT-Regeln), muss man die Regeln vielleicht noch etwas enger formulieren.

*Beispiel 2:* Unser Proxy namens `proxy` steht in `IP_NET_1`, lauscht auf Port 3128 und soll nur für Clients aus `IP_NET_1` wirksam werden. `IP_NET_1` ist über `IP_NET_1_DEV` erreichbar. Pakete aus weiteren Netzen sollen nicht berücksichtigt werden.

```
...
PF_PREROUTING_x='if:IP_NET_1_DEV:any !@proxy 80 DNAT:@proxy:3128'
    # Anfragen an den HTTP-Port, die nicht vom Proxy, aber über eine
    # interne Schnittstelle (IP_NET_1_DEV) kommen, an den Proxy-Port umleiten.
    # An dieser Stelle ist es wichtig, mit if:IP_NET_1_DEV:any zu
    # überprüfen, ob die Pakete von innen kommen, da sonst auch Pakete von
    # außen umgeleitet würden (Sicherheitslücke!).

PF_POSTROUTING_x='prot:tcp IP_NET_1 @proxy:3128 SNAT:IP_NET_1_IPADDR'
    # HTTP-Pakete die aus IP_NET_1 stammen und für den Proxy-Port 3128
    # gedacht sind, so umschreiben, als wären sie vom fli4l (IP_NET_1_IPADDR)

PF_FORWARD_x='prot:tcp @proxy 80 ACCEPT'
    # HTTP-Pakete vom Proxy durch die FORWARD-Kette durchlassen (wenn nötig)
...
```

*Beispiel 3:* Um sich das Leben etwas zu erleichtern und die Regeln kürzer zu gestalten, kann man auch Templates einsetzen (vgl. [Templates im Paketfilter](#) (Seite 50)). Zweckmäßig ist an dieser Stelle das `tmpl:http`, das in `prot:tcp any any:80` übersetzt wird. So wird z. B. aus `tmpl:http IP_NET_1 DNAT:@proxy:3128` dann `prot:tcp IP_NET_1 80 DNAT:@proxy:3128`.

Sowohl `IP_NET_1` als auch `IP_NET_2` sollen transparent über den Proxy umgeleitet werden. Damit ließe sich vereinfacht auch schreiben:

```
...
```

### 3. Basiskonfiguration

```
PF_PREROUTING_x='tmpl:http @proxy ACCEPT'
# HTTP-Pakete vom Proxy sollen nicht umgeleitet werden

PF_PREROUTING_x='tmpl:http IP_NET_1 DNAT:@proxy:3128'
# HTTP-Pakete aus IP_NET_1 sollen umgeleitet werden

PF_PREROUTING_x='tmpl:http IP_NET_2 DNAT:@proxy:3128'
# HTTP-Pakete aus IP_NET_2 sollen umgeleitet werden

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'
...
```

Und so ließe sich das endlos fortsetzen ...

#### 3.13.7. DMZ – Demilitarisierte Zone

fi4l gestattet auch den Aufbau einer DMZ. Hier sei erstmal auf das Wiki verwiesen. <https://ssl.networks.org/wiki>

#### 3.13.8. Conntrack-Helfer

Die Verwendung von IP-Masquerading hat zwar den Vorteil, dass mehrere Rechner im LAN über eine einzige offizielle IP-Adresse geroutet werden kann, es gibt aber auch Nachteile, die man in Kauf nehmen muss.

Ein großes Problem ist zum Beispiel, dass kein Rechner von außen von sich aus eine Verbindung zu einem Rechner aufnehmen kann. Das ist zwar aus Sicherheitsgründen eigentlich durchaus erwünscht, aber bestimmte Protokolle funktionieren nicht mehr, weil sie einen Verbindungsaufbau von außen einfach erfordern.

Ein klassisches Beispiel ist FTP. Neben dem Kommunikationskanal, auf dem Befehle und Antworten ausgetauscht werden, wird ein weiterer Kanal (in Form eines IP-Ports) verwendet, um die eigentlichen Nutzdaten zu versenden. fi4l verwendet dafür bestimmte Conntrack-Helfer, um solche zusätzlichen Ports, die verwendet werden, ad hoc dann freizuschalten und an den internen Rechner weiterzuleiten, wenn sie benötigt werden. Dabei “horcht” der Conntrack-Helfer in den Datenstrom, um zu erkennen, wann ein zusätzlicher Port benötigt wird.

Typische Anwendungen für Conntrack-Helfer sind Chat-Protokolle und Spiele im Internet.

Ein solcher Conntrack-Helfer wird über Regeln in zwei speziellen Arrays aktiviert. Das Array `PF_PREROUTING_CT_%` enthält Helfer-Zuordnungen zu Paketen, die von außen kommen, das Array `PF_OUTPUT_CT_%` enthält Helfer-Zuordnungen zu Paketen, die auf dem Router generiert werden. Einige Beispiele aus der Praxis sollen dies verdeutlichen.

*Beispiel 1:* Soll aktives FTP aus dem LAN erlaubt werden, ist das aus der Sicht des Routers eine Verbindung von außerhalb, somit muss ein Eintrag in `PF_PREROUTING_CT_%` vorgenommen werden:

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

### 3. Basiskonfiguration

Damit wird für alle TCP-Verbindungen aus dem lokalen Netz (`IP_NET_1`) zu irgendeiner anderen Adresse an Port 21 (dies ist der `ftp`-Port) das `ftp`-Hilfsmodul geladen. Dieses Modul erlaubt dann im Laufe der Verbindung, dass der FTP-Server zurück zum Client eine Datenverbindung aufbauen kann, indem temporär ein “Loch” in der Firewall aufgemacht wird.

*Beispiel 2:* Soll passives FTP für einen FTP-Server im LAN ermöglicht werden (dabei wird die Datenverbindung von außen nach innen aufgebaut, so dass auch hier kurzfristig ein Loch in der Firewall geöffnet werden muss), ist dies ebenfalls aus der Sicht des Router eine Verbindung von außerhalb des Routers. Hier sieht die Regel folgendermaßen aus:

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

Mit dieser Regel wird ausgedrückt, dass alle FTP-Verbindungen, die an die dynamische Adresse des Routers gesandt werden, mit dem FTP-Conntrack-Helfer assoziiert werden. Hier wurde `dynamic` verwendet, da angenommen wird, dass der Router für die Einwahl ins Internet verantwortlich ist und somit eine externe IP-Adresse besitzt. Falls der Router eine Einwahl via DSL durchführt, kann man die Regel auch so schreiben:

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

Mit dieser Regel wird ausgedrückt, dass alle FTP-Verbindungen, die von der DSL-Schnittstelle (`pppoe`) kommen, mit dem FTP-Conntrack-Helfer assoziiert werden.

Falls der Router sich nicht einwählt, sondern z. B. hinter einem anderen Router (Fritz!Box, Kabelmodem etc.) hängt, so kann die folgende Regel verwendet werden:

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

Dabei wird im Beispiel angenommen, dass die Verbindung zum anderen Router über die Schnittstelle durchgeführt wird, die dem zweiten Subnetz zugeordnet ist (`IP_NET_2_DEV`).

Zu beachten ist, dass natürlich *zusätzlich* eine entsprechende Konfiguration der FORWARD-Kette nötig ist, um die FTP-Pakete auch tatsächlich weiterzuleiten. Eine typische Regel wäre etwa

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

wobei angenommen wird, dass der Host, auf dem das FTP-Serverprogramm läuft, den Namen `ftpserver` hat.

*Beispiel 3:* Schließlich muss auch, wenn man vom fl4l direkt aktives FTP benutzen möchte (etwa mit Hilfe des `ftp`-Programms aus dem `tools`-Paket), die Firewall dafür vorbereitet werden, diesmal in der OUTPUT-Kette, die mit Hilfe des Arrays `PF_OUTPUT_CT_%` konfiguriert wird:

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

Diese Regel ist jedoch unnötig, falls `FTP_PF_ENABLE_ACTIVE='yes'` benutzt wird – siehe hierzu die Dokumentation des `ftp`-OPTs im `tools`-Paket.

Es folgt eine Übersicht über die existierenden Conntrack-Helfer:

Helfer	Erläuterung
ftp	File Transfer Protocol
h323	H.323 (Voice over IP)
irc	Internet Relay Chat
pptp	PPTP Masquerading (Mit diesem Modul lässt sich mehr als ein PPTP-Client gleichzeitig hinter einem fli4l-Router betreiben.)
sip	Session Initiation Protocol
sane	SANE Network Procotol
snmp	Simple Network Management Protocol
tftp	Trivial File Transfer Protocol

Tabelle 3.8.: Verfügbare Conntrack-Helfer im Paketfilter

Es folgt eine Übersicht der zu konfigurierenden Variablen:

**PF\_PREROUTING\_CT\_ACCEPT\_DEF** Steht diese Variable auf ‘yes’, werden Standard-Regeln generiert, die für ein korrektes Funktionieren des Routers notwendig sind. Standardmäßig sollte man hier ‘yes’ eintragen.

**PF\_PREROUTING\_CT\_N PF\_PREROUTING\_CT\_x PF\_PREROUTING\_CT\_x\_COMMENT**  
Liste der Regeln, die beschreiben, welche eingehenden Pakete vom Router mit Conntrack-Helfern verbunden werden.

**PF\_OUTPUT\_CT\_ACCEPT\_DEF** Steht diese Variable auf ‘yes’, werden Standard-Regeln generiert, die für ein korrektes Funktionieren des Routers notwendig sind. Standardmäßig sollte man hier ‘yes’ eintragen.

**PF\_OUTPUT\_CT\_N PF\_OUTPUT\_CT\_x PF\_OUTPUT\_CT\_x\_COMMENT**  
Liste der Regeln, die beschreiben, welche auf dem Router generierten Pakete vom Router mit Conntrack-Helfern verbunden werden.

### 3.14. Der Paketfilter (IPv6)

Wie für IPv4 wird auch für IPv6-Netzwerke eine Firewall benötigt, damit nicht jeder von außen jeden Rechner im lokalen Netz erreichen kann. Dies ist um so wichtiger, als dass jeder Rechner im Normalfall eine weltweit eindeutige IPv6-Adresse erhält, die dem Rechner permanent zugeordnet werden kann, da sie auf der MAC-Adresse der verwendeten Netzwerkkarte aufbaut.<sup>8</sup> Deshalb verbietet die Firewall erst einmal jegliche Zugriffe von außen und kann dann durch entsprechende Einträge in diesem Abschnitt Stück für Stück – je nach Bedarf – geöffnet werden.

Die Konfiguration der IPv6-Firewall entspricht im Großen und Ganzen der Konfiguration der IPv4-Firewall. Auf Besonderheiten und Unterschiede wird gesondert eingegangen.

<sup>8</sup>Eine Ausnahme existiert, wenn auf den LAN-Hosts die so genannten “Privacy Extensions” aktiviert werden, weil dann ein Teil der IPv6-Adresse zufällig generiert wird. Diese Adressen sind jedoch per Definition nicht nach außen hin bekannt und somit für die Firewall-Konfiguration nur bedingt bis gar nicht relevant.

**PF6\_LOG\_LEVEL** Für alle folgenden Ketten gilt die in PF6\_LOG\_LEVEL vorgenommene Einstellung der Protokoll-Stufe, deren Inhalt auf einen der folgenden Werte gesetzt werden kann: debug, info, notice, warning, err, crit, alert, emerg.

**PF6\_INPUT\_POLICY** Diese Variable legt die Standard-Strategie für auf dem Router eingehende Pakete fest (INPUT-Kette). Mögliche Werte sind "REJECT" (Standard, weist alle Pakete ab), "DROP" (verwirft klammheimlich alle Pakete) und "ACCEPT" (akzeptiert alle Pakete). Für eine genauere Beschreibung siehe die Dokumentation der Variable PF\_INPUT\_POLICY.

Standard-Einstellung: PF6\_INPUT\_POLICY='REJECT'

**PF6\_INPUT\_ACCEPT\_DEF** Diese Variable aktiviert die voreingestellten Regeln für die INPUT-Kette der IPv6-Firewall. Mögliche Werte sind "yes" und "no".

Die voreingestellten Regeln öffnen die Firewall für eingehende ICMPv6-Pings (ein Ping pro Sekunde als Limit) sowie für NDP-Pakete (Neighbour Discovery Protocol), das zur zustandslosen Selbstkonfiguration von IPv6-Netzen benötigt wird. Verbindungen von localhost sowie Antwortpakete zu lokal initiierten Verbindungen werden ebenfalls erlaubt. Schließlich wird die IPv4-Firewall dahingehend angepasst, dass für jeden Tunnel gekapselte IPv6-in-IPv4-Pakete vom Tunnelendpunkt akzeptiert werden.

Standard-Einstellung: PF6\_INPUT\_ACCEPT\_DEF='yes'

**PF6\_INPUT\_LOG** Diese Variable aktiviert das Logging aller zurückgewiesenen eingehenden Pakete. Mögliche Werte sind "yes" und "no". Für eine genauere Beschreibung siehe die Dokumentation der Variable PF\_INPUT\_LOG.

Standard-Einstellung: PF6\_INPUT\_LOG='no'

**PF6\_INPUT\_LOG\_LIMIT** Diese Variable konfiguriert das Log-Limit der INPUT-Kette der IPv6-Firewall, um die Log-Datei lesbar zu halten. Für eine genauere Beschreibung siehe die Dokumentation der Variable PF\_INPUT\_LOG\_LIMIT.

Standard-Einstellung: PF6\_INPUT\_LOG\_LIMIT='3/minute:5'

**PF6\_INPUT\_REJ\_LIMIT** Diese Variable stellt das Limit für das Zurückweisen von eingehenden TCP-Paketen ein. Überschreitet ein solches Paket dieses Limit, wird das Paket klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable PF\_INPUT\_REJ\_LIMIT.

Standard-Einstellung: PF6\_INPUT\_REJ\_LIMIT='1/second:5'

**PF6\_INPUT\_UDP\_REJ\_LIMIT** Diese Variable stellt das Limit für das Zurückweisen von eingehenden UDP-Paketen ein. Überschreitet ein solches UDP-Paket dieses Limit, wird es klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable PF\_INPUT\_UDP\_REJ\_LIMIT.

Standard-Einstellung: PF6\_INPUT\_UDP\_REJ\_LIMIT='1/second:5'

**PF6\_INPUT\_ICMP\_ECHO\_REQ\_LIMIT** Definiert, wie häufig auf eine ICMPv6-Echo-Anfrage reagiert werden soll. Die Häufigkeit wird analog zur Limit-Einschränkung als 'n/Zeiteinheit' mit Bursts beschrieben, also z.B. '3/minute:5'. Ist das Limit überschritten, wird das



### 3. Basiskonfiguration

Paket einfach ignoriert (DROP). Ist dieser Eintrag leer, wird der Standardwert '1/second:5' verwendet; enthält er 'none', wird keine Limitierung durchgeführt.

Standard-Einstellung: `PF6_INPUT_ICMP_ECHO_REQ_LIMIT='1/second:5'`

**PF6\_INPUT\_ICMP\_ECHO\_REQ\_SIZE** Definiert, wie groß eine empfangene ICMPv6-Echo-Anfrage sein darf (in Bytes). In dieser Angabe sind neben den "Nutzdaten" auch die Paket-Header mit zu berücksichtigen. Der Standard-Wert liegt bei 150 Bytes.

Standard-Einstellung: `PF6_INPUT_ICMP_ECHO_REQ_SIZE='150'`

**PF6\_INPUT\_N** Diese Variable enthält die Anzahl der IPv6-Firewallregeln für eingehende Pakete (INPUT-Kette). Standardmäßig werden zwei Regeln aktiviert: Die erste erlaubt allen lokalen Hosts Zugriff auf den Router über so genannte Link-Level-Adressen, und die zweite erlaubt die Kommunikation von Hosts aus dem ersten definierten IPv6-Subnetz mit dem Router.

Falls mehrere lokale IPv6-Subnetze definiert werden, muss die zweite Regel entsprechend oft vervielfältigt werden. Siehe hierzu die Konfigurationsdatei.

Beispiel: `PF6_INPUT_N='2'`

**PF6\_INPUT\_x** Diese Variable spezifiziert eine Regel für die INPUT-Kette der IPv6-Firewall. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_INPUT_x`.

Unterschiede zur IPv4-Firewall:

- Anstatt `IP_NET_x` wird hier `IPV6_NET_x` benutzt.
- Anstatt `IP_ROUTE_x` wird hier `IPV6_ROUTE_x` benutzt.
- IPv6-Adressen müssen in eckigen Klammern eingeschlossen werden (inklusive der Netzmaske, falls vorhanden).
- Alle IPv6-Adressangaben (also auch `IPV6_NET_x` etc.) müssen in eckigen Klammern eingeschlossen werden, falls ein Port oder ein Portbereich folgt.

Beispiele:

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

**PF6\_INPUT\_x\_COMMENT** Diese Variable enthält eine Beschreibung bzw. einen Kommentar zur zugehörigen INPUT-Regel.

Beispiel: `PF6_INPUT_3_COMMENT='no_samba_traffic_allowed'`

**PF6\_FORWARD\_POLICY** Diese Variable legt die Standard-Strategie für von dem Router weiterzuleitenden Pakete fest (FORWARD-Kette). Mögliche Werte sind "REJECT" (Standard, weist alle Pakete ab), "DROP" (verwirft klammheimlich alle Pakete) und "ACCEPT" (akzeptiert alle Pakete). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_POLICY`.

Standard-Einstellung: `PF6_FORWARD_POLICY='REJECT'`

**PF6\_FORWARD\_ACCEPT\_DEF** Diese Variable aktiviert die voreingestellten Regeln für die FORWARD-Kette der IPv6-Firewall. Mögliche Werte sind “yes” und “no”.

Die voreingestellten Regeln öffnen die Firewall für ausgehende ICMPv6-Pings (ein Ping pro Sekunde als Limit). Antwortpakete zu bereits erlaubten Verbindungen werden ebenfalls erlaubt.

Standard-Einstellung: `PF6_FORWARD_ACCEPT_DEF='yes'`

**PF6\_FORWARD\_LOG** Diese Variable aktiviert das Logging aller zurückgewiesenen weiterzuleitenden Pakete. Mögliche Werte sind “yes” und “no”. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_LOG`.

Standard-Einstellung: `PF6_FORWARD_LOG='no'`

**PF6\_FORWARD\_LOG\_LIMIT** Diese Variable konfiguriert das Log-Limit der FORWARD-Kette der IPv6-Firewall, um die Log-Datei lesbar zu halten. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_LOG_LIMIT`.

Standard-Einstellung: `PF6_FORWARD_LOG_LIMIT='3/minute:5'`

**PF6\_FORWARD\_REJ\_LIMIT** Diese Variable stellt das Limit für das Zurückweisen von weiterzuleitenden TCP-Paketen ein. Überschreitet ein solches TCP-Paket dieses Limit, wird es klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_REJ_LIMIT`.

Standard-Einstellung: `PF6_FORWARD_REJ_LIMIT='1/second:5'`

**PF6\_FORWARD\_UDP\_REJ\_LIMIT** Diese Variable stellt das Limit für das Zurückweisen von weiterzuleitenden UDP-Paketen ein. Überschreitet ein solches UDP-Paket dieses Limit, wird es klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_UDP_REJ_LIMIT`.

Standard-Einstellung: `PF6_FORWARD_UDP_REJ_LIMIT='1/second:5'`

**PF6\_FORWARD\_N** Diese Variable enthält die Anzahl der IPv6-Firewallregeln für weiterzuleitende Pakete (FORWARD-Kette). Standardmäßig werden zwei Regeln aktiviert: Die erste verhindert die Weiterleitung aller lokalen Samba-Pakete in nicht-lokale Netze, und die zweite erlaubt letzteres für alle anderen lokalen Pakete aus dem ersten definierten IPv6-Subnetz.

Falls mehrere lokale IPv6-Subnetze definiert werden, muss die letzte Regel entsprechend oft vervielfältigt werden. Siehe hierzu die Konfigurationsdatei.

Beispiel: `PF6_FORWARD_N='2'`

**PF6\_FORWARD\_x** Diese Variable spezifiziert eine Regel für die FORWARD-Kette der IPv6-Firewall. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_FORWARD_x`.

Unterschiede zur IPv4-Firewall:

- Anstatt `IP_NET_x` wird hier `IPV6_NET_x` benutzt.
- Anstatt `IP_ROUTE_x` wird hier `IPV6_ROUTE_x` benutzt.
- IPv6-Adressen müssen in eckigen Klammern eingeschlossen werden (inklusive der Netzmaske, falls vorhanden).

### 3. Basiskonfiguration

- Alle IPv6-Adressangaben (also auch IPV6\_NET\_x etc.) müssen in eckigen Klammern eingeschlossen werden, falls ein Port oder ein Portbereich folgt.

Beispiele:

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

**PF6\_FORWARD\_x\_COMMENT** Diese Variable enthält eine Beschreibung bzw. einen Kommentar zur zugehörigen FORWARD-Regel.

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

**PF6\_OUTPUT\_POLICY** Diese Variable legt die Standard-Strategie für vom Router ausgehende Pakete fest (OUTPUT-Kette). Mögliche Werte sind "REJECT" (Standard, weist alle Pakete ab), "DROP" (verwirft klammheimlich alle Pakete) und "ACCEPT" (akzeptiert alle Pakete). Für eine genauere Beschreibung siehe die Dokumentation der Variable PF\_OUTPUT\_POLICY.

Standard-Einstellung: PF6\_OUTPUT\_POLICY='REJECT'

**PF6\_OUTPUT\_ACCEPT\_DEF** Diese Variable aktiviert die voreingestellten Regeln für die OUTPUT-Kette der IPv6-Firewall. Mögliche Werte sind "yes" und "no". Momentan existieren keine voreingestellten Regeln.

Standard-Einstellung: PF6\_OUTPUT\_ACCEPT\_DEF='yes'

**PF6\_OUTPUT\_LOG** Diese Variable aktiviert das Logging aller zurückgewiesenen ausgehenden Pakete. Mögliche Werte sind "yes" und "no". Für eine genauere Beschreibung siehe die Dokumentation der Variable PF\_OUTPUT\_LOG.

Standard-Einstellung: PF6\_OUTPUT\_LOG='no'

**PF6\_OUTPUT\_LOG\_LIMIT** Diese Variable konfiguriert das Log-Limit der OUTPUT-Kette der IPv6-Firewall, um die Log-Datei lesbar zu halten. Für eine genauere Beschreibung siehe die Dokumentation der Variable PF\_OUTPUT\_LOG\_LIMIT.

Standard-Einstellung: PF6\_OUTPUT\_LOG\_LIMIT='3/minute:5'

**PF6\_OUTPUT\_REJ\_LIMIT** Diese Variable stellt das Limit für das Zurückweisen von ausgehenden TCP-Paketen ein. Überschreitet ein solches Paket dieses Limit, wird das Paket klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable PF\_OUTPUT\_REJ\_LIMIT.

Standard-Einstellung: PF6\_OUTPUT\_REJ\_LIMIT='1/second:5'

**PF6\_OUTPUT\_UDP\_REJ\_LIMIT** Diese Variable stellt das Limit für das Zurückweisen von ausgehenden UDP-Paketen ein. Überschreitet ein solches UDP-Paket dieses Limit, wird es klammheimlich verworfen (DROP). Für eine genauere Beschreibung siehe die Dokumentation der Variable PF\_OUTPUT\_UDP\_REJ\_LIMIT.

Standard-Einstellung: PF6\_OUTPUT\_UDP\_REJ\_LIMIT='1/second:5'

**PF6\_OUTPUT\_N** Diese Variable enthält die Anzahl der IPv6-Firewallregeln für eingehende Pakete (OUTPUT-Kette). Standardmäßig werden zwei Regeln aktiviert: Die erste erlaubt allen lokalen Hosts Zugriff auf den Router über so genannte Link-Level-Adressen, und die zweite erlaubt die Kommunikation von Hosts aus dem ersten definierten IPv6-Subnetz mit dem Router.

Falls mehrere lokale IPv6-Subnetze definiert werden, muss die zweite Regel entsprechend oft vervielfältigt werden. Siehe hierzu die Konfigurationsdatei.

Beispiel: `PF6_OUTPUT_N='1'`

**PF6\_OUTPUT\_x** Diese Variable spezifiziert eine Regel für die OUTPUT-Kette der IPv6-Firewall. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_OUTPUT_x`.

Unterschiede zur IPv4-Firewall:

- Anstatt `IP_NET_x` wird hier `IPV6_NET_x` benutzt.
- Anstatt `IP_ROUTE_x` wird hier `IPV6_ROUTE_x` benutzt.
- IPv6-Adressen müssen in eckigen Klammern eingeschlossen werden (inklusive der Netzmaske, falls vorhanden).
- Alle IPv6-Adressangaben (also auch `IPV6_NET_x` etc.) müssen in eckigen Klammern eingeschlossen werden, falls ein Port oder ein Portbereich folgt.

Beispiele:

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

**PF6\_OUTPUT\_x\_COMMENT** Diese Variable enthält eine Beschreibung bzw. einen Kommentar zur zugehörigen OUTPUT-Regel.

Beispiel: `PF6_OUTPUT_3_COMMENT='no_samba_traffic_allowed'`

**PF6\_USR\_CHAIN\_N** Diese Variable enthält die Anzahl der vom Benutzer definierten IPv6-Firewall-Tabellen. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_USR_CHAIN_N`.

Standard-Einstellung: `PF6_USR_CHAIN_N='0'`

**PF6\_USR\_CHAIN\_x\_NAME** Diese Variable enthält den Namen der entsprechenden benutzerdefinierten IPv6-Firewall-Tabelle. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_USR_CHAIN_x_NAME`.

Beispiel: `PF6_USR_CHAIN_1_NAME='usr-myvpn'`

**PF6\_USR\_CHAIN\_x\_RULE\_N** Diese Variable enthält die Anzahl der IPv6-Firewallregeln in der zugehörigen benutzerdefinierten IPv6-Firewall-Tabelle. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_USR_CHAIN_x_RULE_N`.

Beispiel: `PF6_USR_CHAIN_1_RULE_N='0'`

**PF6\_USR\_CHAIN\_x\_RULE\_x** Diese Variable spezifiziert eine Regel für die benutzerdefinierte IPv6-Firewall-Tabelle. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_USR_CHAIN_x_RULE_x`.

Unterschiede zur IPv4-Firewall:

- Anstatt `IP_NET_x` wird hier `IPV6_NET_x` benutzt.
- Anstatt `IP_ROUTE_x` wird hier `IPV6_ROUTE_x` benutzt.
- IPv6-Adressen müssen in eckigen Klammern eingeschlossen werden (inklusive der Netzmaske, falls vorhanden).
- Alle IPv6-Adressangaben (also auch `IPV6_NET_x` etc.) müssen in eckigen Klammern eingeschlossen werden, falls ein Port oder ein Portbereich folgt.

**PF6\_USR\_CHAIN\_x\_RULE\_x\_COMMENT** Diese Variable enthält eine Beschreibung bzw. einen Kommentar zur zugehörigen Regel.

Beispiel: `PF6_USR_CHAIN_1_RULE_1_COMMENT='some_user-defined_rule'`

**PF6\_POSTROUTING\_N** Diese Variable enthält die Anzahl der IPv6-Firewallregeln fürs Maskieren (POSTROUTING-Kette). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_POSTROUTING_N`.

Beispiel: `PF6_POSTROUTING_N='2'`

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

Eine Liste der Regeln, die beschreiben, welche IPv6-Pakete vom Router maskiert werden (bzw. unmaskiert weitergeleitet werden). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_POSTROUTING_x`.

**PF6\_PREROUTING\_N** Diese Variable enthält die Anzahl der IPv6-Firewallregeln fürs Weiterleiten an ein anderes Ziel (PREROUTING-Kette). Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_PREROUTING_N`.

Beispiel: `PF6_PREROUTING_N='2'`

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

Eine Liste der Regeln, die beschreiben, welche IPv6-Pakete vom Router an ein anderes Ziel weitergeleitet werden sollen. Für eine genauere Beschreibung siehe die Dokumentation der Variable `PF_PREROUTING_x`.

## 3.15. Domain-Konfiguration

Windows-PCs im LAN haben eine unangenehme Eigenschaft: Sobald ein Nameserver benötigt wird und man diesen deshalb im Windows einstellt, fragen diese Windows-Rechner den angegebenen Nameserver in regelmäßigen Abständen ab – auch wenn man gar nicht daran arbeitet! Würde man also auf dem Windows-PC einen DNS-Server im Internet angeben, wird's teuer...

Der Trick ist nun folgender: Wenn im LAN nicht bereits ein DNS-Server vorhanden ist, kann man den DNS-Server im fli4l-Router verwenden.

Es wird `DNSMASQ` als DNS-Server eingesetzt.

Wenn man jedoch mit der DNS-Konfiguration beginnt, sollte man sich zunächst Gedanken über den Domain-Namen und die Namen der PCs im Netz machen. Der verwendete Domain-Name wird nicht im Internet sichtbar. Deshalb kann man sich hier prinzipiell beliebige Domain-Namen ausdenken.

Außerdem sollte man jedem Windows-Rechner einen Namen verpassen. Diese Namen müssen dem fli4l-Router bekannt sein.

**DOMAIN\_NAME** Standard-Einstellung: `DOMAIN_NAME='lan.fli4l'`

Hier kann sich jeder austoben, da die lokal verwendete Domain nicht im Internet sichtbar wird. Sie sollten lediglich vermeiden, einen Namen zu benutzen, den es im Internet geben könnte (z.B. *irgendwas.de*), da Sie sonst nicht auf diese Domain werden zugreifen können.

**DNS\_FORWARDERS** Standard-Einstellung: `DNS_FORWARDERS=""`

Hier ist die DNS-Server-Adresse des Internet-Providers anzugeben, wenn fli4l als Router in das Internet verwendet wird. Der fli4l-Router gibt dann sämtliche DNS-Anfragen, die er nicht selbst beantworten kann, an diese Adresse weiter.

Möchte man mehrere DNS-Forwarder angeben, trennt man die IP-Adressen durch Leerzeichen.

Sind mehrere DNS-Server konfiguriert werden diese in der angegebenen Reihenfolge für DNS-Anfragen genutzt, somit wird der zweite angegebene Server nur genutzt, wenn der erste keine Antwort liefert usw.

Es ist auch möglich, optional Port-Nummern an die IP-Adressen durch Doppelpunkt getrennt anzugeben. Allerdings muss dann `OPT_DNS='yes'` (Seite ??) sein (Paket `dns_dhcp` (Seite ??)) und es darf nirgends die Option `*_USEPEERDNS` benutzt werden.

Achtung: Auch wenn

- `PPPOE_USEPEERDNS` (Seite ??),
- `ISDN_CIRC_x_USEPEERDNS` (Seite ??) oder
- `DHCP_CLIENT_x_USEPEERDNS` (Seite ??)

gesetzt (`=yes`) ist, ist hier die Eintragung eines Servers nötig, da sonst direkt nach dem Start keine Namensauflösung möglich ist.

Ausnahme: fli4l als Router in einem lokalen Netz *ohne* Anschluss an das Internet oder (Firmen-)Netze mit weiteren DNS-Servern. In diesem Fall ist `127.0.0.1` anzugeben, um das Weiterleiten zu unterbinden.

**HOSTNAME\_IP** (optional)

Hiermit kann optional festgelegt werden, an welches Netz `IP_NET_x` der `HOSTNAME` gebunden wird.

**HOSTNAME\_ALIAS\_N** (optional)

Anzahl der zusätzlichen Alias-Hostnamen für den Router.

**HOSTNAME\_ALIAS\_x** (optional)

Zusätzlicher Alias-Name für den Router.

## 3.16. imond-Konfiguration

**OPT\_IMOND** Standard-Einstellung: `OPT_IMOND='no'`

Mit `OPT_IMOND` kann man einstellen, ob der imond-Server aktiviert werden soll. imond übernimmt dabei das Monitoring/Controlling und Least-Cost-Routing des fli4l-Routers. Der [Beschreibung von imond](#) (Seite 120) ist deshalb ein extra Kapitel gewidmet (s.u.).

### 3. Basiskonfiguration

Wichtig: Die LC-Routing-Features von fli4l können nur mit imond genutzt werden. Ein zeitabhängiges Umschalten von Verbindungen ist ohne imond nicht möglich!

Für ISDN- und DSL-Routing ist imond ab Version 1.5 zwingend erforderlich. In diesem Fall ist `OPT_IMOND='yes'` einzustellen.

Wird fli4l lediglich als Router zwischen 2 Netzwerken eingesetzt, sollte `OPT_IMOND='no'` eingestellt werden.

**IMOND\_PORT** TCP/IP-Port, auf dem imond auf Verbindungen horcht. Der Standard-Wert '5000' sollte nur in Ausnahmefällen geändert werden.

**IMOND\_PASS** Standard-Einstellung: `IMOND_PASS=""`

Hier kann ein spezielles User-Password für imond gesetzt werden. Meldet sich ein Client auf Port 5000 an, erwartet imond (und damit auch seine Clients) die Eingabe dieses Passwords, bevor er irgendeinen Befehl korrekt beantwortet. Ausnahme: Befehle "quit", "help" und "pass". Ist `IMOND_PASS` leer, wird kein Password benötigt.

Ob der Client im User-Modus bestimmte Steuerbefehle, wie Dial, Hangup, Reboot, Umschalten der Default-Route bereits ausführen kann oder dafür die Eingabe des Admin-Passwords zwingend notwendig ist, wird über die Variablen

- [IMOND\\_ENABLE](#) (Seite 81),
- [IMOND\\_DIAL](#) (Seite 80),
- [IMOND\\_ROUTE](#) (Seite 80) und
- [IMOND\\_REBOOT](#) (Seite 80)

eingestellt, siehe unten.

**IMOND\_ADMIN\_PASS** Standard-Einstellung: `IMOND_ADMIN_PASS=""`

Mit Hilfe der Admin-Passwords erhält der Client alle Rechte und kann so sämtliche Steuerfunktionen des imond-Servers nutzen – und zwar unabhängig von den Variablen `IMOND_ENABLE`, `IMOND_DIAL` usw. Lässt man `IMOND_ADMIN_PASS` leer, so reicht die Eingabe des User-Passwords, um sämtliche Rechte zu erhalten!

**IMOND\_LED** imond kann den Online/Offline-Status nun über eine LED anzeigen. Diese wird folgendermaßen an einen COM-Port angeschlossen:

Verbindung 25-polig:

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

Verbindung 9-polig:

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

Ist eine ISDN- oder DSL-Verbindung aufgebaut, leuchtet die LED. Ansonsten ist sie ausgeschaltet. Sollte es genau umgekehrt sein, ist die Leuchtdiode umzupolen. Sollte die LED zu schwach leuchten, kann der Vorwiderstand bis auf 470 Ohm reduziert werden.

Es ist auch möglich, zwei verschiedenfarbige LEDs anzuschließen. Dann ist die zweite LED ebenso über einen Vorwiderstand zwischen DTR und GND anzuschließen, jedoch

### 3. Basiskonfiguration

genau umgekehrt. Dann leuchtet je nach Zustand die eine oder die andere LED. Oder man verwendet direkt eine DUO-LED (zweifarbige, drei Anschlussbeinchen).

Im Moment verhält sich der RTS-Anschluss der seriellen Schnittstelle genauso wie DTR. Hier könnte also noch eine weitere LED angeschlossen werden, die den Online/Offline-Zustand anzeigt. Das könnte sich jedoch in einer zukünftigen fli4l-Version ändern.

Als Wert von `IMOND_LED` muss ein COM-Port angegeben werden, also `'com1'`, `'com2'`, `'com3'` oder `'com4'`. Ist keine LED angeschlossen, sollte die Variable leer gelassen werden.

**IMOND\_BEEP** Mit `IMOND_BEEP='yes'` gibt imond einen Zweiklang-Ton über den PC-Lautsprecher aus, wenn der Zustand von Offline nach Online wechselt und umgekehrt. Im ersten Fall wird zuerst ein tiefer, dann ein hoher Ton ausgegeben. Beim Wechsel in den Offline-Status zurück wird zuerst der höhere, dann der tiefere Ton ausgegeben.

**IMOND\_LOG** Standard-Einstellung: `IMOND_LOG='no'`

Wird `IMOND_LOG='yes'` benutzt, werden in der Datei `/var/log/imond.log` die Verbindungen protokolliert. Diese Datei kann z.B. für Statistikzwecke per scp auf einen Rechner im LAN kopiert werden. Für den scp-Zugriff ist aber dann noch das Paket sshd zu installieren und so zu konfigurieren, dass es auch scp zur Verfügung stellt.

Das Format der Logdateieinträge ist in Tabelle 3.9 beschrieben.

Tabelle 3.9.: Format der Imond-Logdatei

Eintrag	Bedeutung
Circuit	der Name des Circuits, für den der Eintrag erzeugt wurde
Startzeit	Datum und Uhrzeit der Einwahl dieses Circuits
Stopzeit	Datum und Uhrzeit des Auflegens dieses Circuits
Online-Zeit	die Zeit, die dieser Circuit online war
Abgerechnete Zeit	die Zeit, die der Provider abrechnen wird (hängt vom Takt ab)
Kosten	die Kosten, die der Provider für die Zeit in Rechnung stellt
Bandbreite	die genutzte Bandbreite getrennt nach in und out (in zuerst), dargestellt als zwei vorzeichenlose Integerzahlen, für die gilt: Bandbreite = $4\text{GiB} * \text{<erste Zahl>} + \text{<zweite Zahl>}$
Device	das Gerät, über das kommuniziert wurde
Abrechnungstakt	der Takt, der vom Provider zur Abrechnung herangezogen wird (Daten der Circuit-Konfiguration)
Taktgebühren	die Gebühren, die pro Takt fällig werden (Daten der Circuit-Konfiguration)

Die Kosten werden in Euro ausgegeben. Wichtig ist dabei die korrekte Definition der entsprechenden Circuit-Variablen `ISDN_CIRC_x_TIMES` (Seite ??).

**IMOND\_LOGDIR** Ist das Protokollieren eingeschaltet, kann über `IMOND_LOGDIR` ein alternatives Verzeichnis statt `/var/log` angegeben werden, z.B. `'/boot'`. Dann wird die Log-Datei `imond.log` auf dem Bootmedium angelegt. Dazu muss dieses aber auch Read/Write „gemounted“ sein. Default ist `'auto'` was den Speicherort automatisch bestimmt. Je nach weiterer Konfiguration liegt das dann unter `/boot/persistent/base` oder an einem anderen durch `FLI4L_UUID` bestimmten Pfad. Ist `/boot` nicht Read/Write und `FLI4L_UUID` nicht gesetzt, befindet sich das File unter `/var/run`.



**IMOND\_ENABLE IMOND\_DIAL IMOND\_ROUTE IMOND\_REBOOT** Durch diese Variablen werden bestimmte Kommandos, die von imonc-Clients zum imond-Server gesendet werden, bereits im User-Modus freigeschaltet.

Hiermit kann man einstellen, ob der imond-Server die ISDN-Schnittstelle ein- bzw. ausschalten, wählen/einhängen, eine neue Default-Route setzen und/oder den Rechner booten darf.

Standard-Einstellungen:

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

Alle weiteren Features der Client-/Server-Schnittstelle von imond sind in einem [eigenen Kapitel](#) (Seite 120) beschrieben.

## 3.17. Circuit-Konfiguration

### 3.17.1. Circuits allgemein

Der fli4l-Router erlaubt ab Version 4.0, Verbindungen nach “außen” flexibel über so genannte “Circuits” zu konfigurieren. Der Begriff “Circuit” kommt aus dem Englischen und bedeutet in diesem Zusammenhang so viel wie “Leitung”. Seine Verwendung in fli4l entstammt dem Wählen und Auflegen von ISDN-Verbindungen, das seit der ersten fli4l-Version möglich ist; da ISDN ein leitungsvermittelnder (“circuit-switched”) Dienst ist, hat sich der Begriff etabliert und wird heute in allen anderen Verbindungs-Situationen verwendet, auch wenn es meistens nicht mehr um leitungsvermittelnde, sondern um paketvermittelnde Dienste geht. Auch in dieser Dokumentation wird der Begriff des Circuits durchgängig verwendet.

Ein konfigurierter Circuit erlaubt es dem fli4l-Router, irgendeine Form der Verbindung zwischen dem Router und einem anderen Netzwerk-Kommunikationspartner herzustellen. Meistens, aber nicht immer, geht es dabei um die Herstellung einer Internet-Anbindung. Im Folgenden wird eine kurze Übersicht darüber gegeben, welche Circuit-Typen der fli4l-Router beherrscht (die meisten davon werden jedoch von anderen Paketen angeboten, dies ist aber entsprechend in der Tabelle vermerkt).

Typ	Paket	OPT	Beschreibung
route	base	-	Mit Hilfe von route-Circuits können Routen in andere Netze konfiguriert werden. Dies entspricht im Wesentlichen der Funktionalität der Variablen IP_ROUTE_% (siehe <a href="#">IP_ROUTE_N</a> (Seite 43)), geht jedoch etwas darüber hinaus. Intern werden alle per IP_ROUTE_% konfigurierte Routen in route-Circuits abgebildet.

### 3. Basiskonfiguration

Typ	Paket	OPT	Beschreibung
dhcp	dhcp_client	OPT_DHCP_CLIENT	Mit Hilfe von dhcp-Circuits lassen sich IPv4- und IPv6-Adressinformationen von einem DHCP-Server ermitteln. Dies ist vor allem dann sinnvoll, wenn der fii4l nicht selbsttätig eine Internet-Verbindung aufbaut, sondern sich hinter einem anderen Router befindet, der dies für einen erledigt, etwa hinter einem Kabelmodem.
isdn	isdn	OPT_ISDN	Mit Hilfe von isdn-Circuits ist eine Einwahl in ein anderes Netz (z. B. ein Firmennetz) über ISDN möglich.
ppp	ppp (+ diverse)	OPT_PPP (+ diverse)	Mit Hilfe von ppp-Circuits ist eine Einwahl ins Internet oder ein Firmennetz über eine Reihe diverser Kanäle mit Hilfe des Point-to-Point-Protokolls (PPP) möglich. Mehr dazu ist in Abschnitt ?? zu finden.

Tabelle 3.10.: Verfügbare Circuit-Typen

Alle Circuits werden der Übersichtlichkeit halber in der Datei `circuits.txt` konfiguriert. Die Anzahl der konfigurierten Circuits wird dabei in der Variable `CIRC_N` festgehalten:

**CIRC\_N** Diese Variable gibt die Anzahl der konfigurierten Circuits an.

Standard-Einstellung: `CIRC_N='0'`

Beispiel: `CIRC_N='4'`

Darüber hinaus besitzt jeder Circuit, egal von welchem Typ, einige allgemeine Attribute. Zuerst operiert *jeder* Circuit auf einer *Netzwerk-Schnittstelle*. Je nach Circuit-Typ ist diese Schnittstelle statisch vorhanden (dies gilt z. B. für Ethernet-Schnittstellen wie “eth0”), oder sie wird dynamisch beim Einwählen erzeugt (das ist z. B. bei den “pppX”-Schnittstellen von ppp-Circuits der Fall), oder sie wird von einem anderen Circuit erzeugt und hier nur referenziert (das ist z. B. bei DHCPv6-over-PPPoE der Fall). Auf Grund dieser verschiedenen Möglichkeiten gibt es keine einheitliche Variable zur Angabe der Schnittstelle, stattdessen kümmert sich jeder Circuit-Typ selbst um die Konfiguration der Schnittstelle (und einige wie ppp verbieten die Angabe einer Schnittstelle gänzlich, da sie automatisch erzeugt und verwaltet wird).

Weitere allgemeine Attribute werden über die folgenden Variablen konfiguriert.

**CIRC\_x\_NAME** Jeder Circuit hat einen Namen. Dieser Name kann aus Buchstaben, Ziffern und dem Bindestrich (‘-’) bestehen und hilft dabei, den Circuit in der Web-GUI, in Protokollen etc. zu identifizieren. Der Name muss unter allen Circuits und Circuit-Klassen (siehe [CIRC\\_CLASS\\_x\\_NAME](#) (Seite 92)) eindeutig sein.

Beispiel: `CIRC_x_NAME='DSL-Telekom'`

**CIRC\_x\_TYPE** Jeder Circuit hat einen Typ. Dieser Typ bestimmt, wie das Wählen und Auflegen funktioniert. Intern entscheidet der Typ, welches Skript zum Wählen und Auflegen verwendet wird.

Es sind immer nur die Typen der aktivierten Pakete verfügbar. Wenn Sie also z. B. einen dhcp-Circuit verwenden möchten, dann müssen Sie das Paket `dhcp_client` herunterladen und DHCP mit `OPT_DHCP_CLIENT='yes'` aktivieren. Andernfalls bekommen Sie bei der Verwendung des Typs "dhcp" beim Bauen der Installationsarchive eine Fehlermeldung.

Beispiel: `CIRC_x_TYPE='dhcp'`

**CIRC\_x\_ENABLED** Die Variable `CIRC_x_ENABLED` aktiviert einen Circuit zur Konfigurationszeit. Damit der Circuit überhaupt Beachtung findet, muss diese Variable auf `'yes'` gesetzt werden. Gilt hingegen `CIRC_x_ENABLED='no'`, dann wird der Circuit auf dem fl4l nicht konfiguriert. Auch kann zur Konfigurationszeit der Circuits nicht mit seinem Namen angesprochen werden, etwa in Firewall-Regeln.

Standard-Einstellung: `CIRC_x_ENABLED='no'`

Beispiel: `CIRC_x_ENABLED='yes'`

**CIRC\_x\_DIALMODE** Jeder Circuit kann einen individuellen initialen [Wählmodus](#) (Seite 90) erhalten, der dann über diese Variable konfiguriert werden kann.

Standard-Einstellung: `CIRC_x_DIALMODE='auto'`

Beispiel: `CIRC_x_DIALMODE='manual'`

**CIRC\_x\_NETS\_IPV4\_y** Ein Circuit ist nur dann sinnvoll, wenn er auch zu einer Netzanbindung führt. Dazu ist es in der Regel notwendig, dass Routen konfiguriert werden, sobald eine erfolgreiche Einwahl stattgefunden hat. Welche IPv4-Netze über den Circuit geroutet werden, kann über diese Variablen festgelegt werden.

Im häufigsten Fall der Circuit-Nutzung, der Internet-Anbindung, muss die Default-Route über den Circuit gehen. Dazu muss das Netz `0.0.0.0/0` in die Liste eingetragen werden.

Nur in Ausnahmefällen müssen keine Netze angegeben werden, etwa bei Server-Circuits, bei denen keine Routen zurück zu den Clients installiert werden sollen.

Standard-Einstellung: `CIRC_x_NETS_IPV4_N='0'`

Beispiel 1:

```
CIRC_x_NETS_IPV4_N='1'
CIRC_x_NETS_IPV4_1='0.0.0.0/0'
```

Beispiel 2:

```
CIRC_x_NETS_IPV4_N='2'
CIRC_x_NETS_IPV4_1='10.15.16.0/24'
CIRC_x_NETS_IPV4_2='10.16.0.0/16'
```

**CIRC\_x\_NETS\_IPV6\_y** Hier werden analog zu `CIRC_x_NETS_IPV4_y` IPv6-Netze angegeben, die nach der Einwahl über den Circuit geroutet werden sollen. Diese Variablen können nur bei aktivierter IPv6-Unterstützung (Paket `ipv6`, `OPT_IPV6='yes'`) verwendet werden.

### 3. Basiskonfiguration

Im häufigsten Fall der Circuit-Nutzung, der Internet-Anbindung, muss die Default-Route über den Circuit gehen. Dazu muss das Netz `::/0` in die Liste eingetragen werden.

Nur in Ausnahmefällen müssen keine Netze angegeben werden, etwa bei Server-Circuits, bei denen keine Routen zurück zu den Clients installiert werden sollen.

Standard-Einstellung: `CIRC_x_NETS_IPV6_N='0'`

Beispiel 1:

```
CIRC_x_NETS_IPV6_N='1'
CIRC_x_NETS_IPV6_1='::/0'
```

Beispiel 2:

```
CIRC_x_NETS_IPV6_N='2'
CIRC_x_NETS_IPV6_1='2001:db8:1::/48'
CIRC_x_NETS_IPV6_2='2001:db8:2::/48'
```

**CIRC\_x\_PROTOCOLS** In der Regel werden die Layer-3-Protokolle, die von einem Circuit unterstützt werden (IPv4 oder IPv6) aus den konfigurierten Netzen (`CIRC_x_NETS_IPV4_y` und `CIRC_x_NETS_IPV6_y`, siehe oben) abgeleitet. In manchen Fällen werden jedoch keine Netze angegeben, weil keine Routen aufgebaut werden sollen. Dies ist beispielsweise bei Server-Circuits der Fall. In solchen Fällen ist es nötig, die zu verwendenden Layer-3-Protokolle explizit einzustellen. Dazu wird in dieser Variable eine Liste von Protokollen notiert, die durch Leerzeichen voneinander getrennt sind. Erlaubte Protokolle sind `ipv4` und `ipv6`.

Standard-Einstellung: abgeleitet aus `CIRC_x_NETS_IPV4_y` und `CIRC_x_NETS_IPV6_y`

Beispiel: `CIRC_x_PROTOCOLS='ipv4 ipv6'`

**CIRC\_x\_UP** Ist `CIRC_x_UP='yes'`, dann wird der entsprechende Circuit beim Booten aktiviert. Je nach [Wählmodus](#) (Seite 90) kann dies bereits beim Booten zu einer Einwahl führen, oder die Einwahl kann später über die GUI oder das `fli4lctrl`-Programm veranlasst werden. Bei `CIRC_x_UP='no'` muss der Circuit erst mit Hilfe der GUI oder mit dem `fli4lctrl`-Programm aktiviert werden, bevor eine Einwahl gestartet werden kann. Falls Circuits aktiviert werden, die sich in einem oder mehreren gerouteten Netzen überlappen, ist dies ein Fehler, der bereits beim Versuch, das Installationsarchiv zu bauen, von `mkfli4l` gemeldet wird.<sup>9</sup>

Diese Einstellung ist nur relevant mit `OPT_CIRCD='no'`.

Standard-Einstellung: `CIRC_x_UP='no'`

Beispiel: `CIRC_x_UP='yes'`

**CIRC\_x\_PRIORITY** Diese Variable gibt die Priorität des Circuits an. Höhere Werte bedeuten eine niedrigere Priorität. Prioritäten dienen dazu, Circuits nach Eignung zu gruppieren: Bei der Auswahl von Circuits<sup>10</sup> werden alle Circuits prioritätsweise abgearbeitet. Nur

---

<sup>9</sup>In einer späteren Version ist angedacht, in einem solchen Fall den Datenverkehr für diese Netze gleichmäßig auf alle diese Circuits zu verteilen.

<sup>10</sup>durch den Hintergrundprozess `circd`

### 3. Basiskonfiguration

wenn in der höchsten Prioritätsklasse keine Circuits in Frage kommen, kommen die Circuits der nächstniedrigeren Prioritätsklasse zum Zuge. Mehr zum Auswahlalgorithmus des `circd` finden Sie im Abschnitt [“Das Programm `circd`”](#) (Seite 95).

Diese Einstellung ist nur relevant mit `OPT_CIRCD='yes'`.

Standard-Einstellung: `CIRC_x_PRIORITY='1'`

Beispiel: `CIRC_x_PRIORITY='2'`

**CIRC\_x\_TIMES** Ist `OPT_CIRCD='yes'`, dann enthält diese Variable eine Zeitspezifikation, die angibt, wann der Circuit aktiviert werden soll und wann nicht, und wie viel der Circuit bei einer erfolgreichen Einwahl pro Minute kostet. Dadurch wird es möglich, zu verschiedenen Zeiten verschiedene Circuits zu verwenden (Least-Cost-Routing). Dabei kontrolliert der Dämon-Prozess `circd` das Aktivieren und Deaktivieren der Circuits.

Der Inhalt der Variablen ist wie folgt aufgebaut:

```
CIRC_x_TIMES='W1-W2:hh-hh:Kosten:Typ [W1-W2:hh-hh:Kosten:Typ [...]]'
```

Jedes Feld besteht aus vier Unterfeldern, die mit Hilfe eines Doppelpunkts (':') voneinander getrennt sind:

- Feld *W1-W2*: Wochentag-Zeitraum, z. B. Mo-Fr oder Sa-Su usw. Sowohl die deutsche als auch die englische Schreibweise sind erlaubt. Soll ein einzelner Wochentag eingetragen werden, ist W1-W1 zu schreiben, also z. B. Su-Su.
- Feld *hh-hh*: Stunden-Bereich, z. B. 09-18 oder auch 18-09. 18-09 ist gleichbedeutend mit 18-24 plus 00-09. 00-24 meint den ganzen Tag. Stunden müssen immer zweistellig (also notfalls mit führenden Nullen) angegeben werden.
- Feld *Kosten*: Hier werden in Euro die Kosten pro Minute angegeben, z. B. 0.032 für 3,2 Cent pro Minute. Diese werden unter Berücksichtigung der Taktzeit in die tatsächlich anfallenden Kosten umgerechnet, welche dann im imon-Client oder der WebGUI angezeigt werden.
- Feld *Typ*: Dieses Feld gibt den Typ des Zeitraums an:
  - Y oder J: Im angegebenen Zeitraum wird der Circuit aktiviert, unabhängig von den anfallenden Kosten.
  - L: Im angegebenen Zeitraum wird der Circuit aktiviert, wenn er zu den günstigsten Circuits gehört. ('L' steht für 'Least cost', also "niedrigste Kosten".)
  - N: Der angegebene Zeitbereich dient nur zum Berechnen von Kosten, der Circuit wird in diesem Zeitraum jedoch nicht aktiviert. Dies kann sinnvoll sein, wenn der Circuit *manuell* aktiviert wird, etwa wenn es sich um einen Circuit zur Verbindung mit einem Firmennetz handelt, der nur bei Bedarf hinzugeschaltet wird.

Der Typ kann weggelassen werden, in diesem Fall wird 'L' angenommen.

Standard-Einstellung: `CIRC_2_TIMES='Mo-Su:00-24:0.0:N'`

Beispiel 1:

```
CIRC_1_TIMES='Mo-Fr:09-18:0.049:N Mo-Fr:18-09:0.044:L Sa-Su:00-24:0.039:Y'
```

### 3. Basiskonfiguration

Beispiel 2 für diejenigen, die eine Flatrate nutzen:

```
CIRC_2_TIMES='Mo-Su:00-24:0.0:Y'
```

**Wichtig:** Wenn die Zeitbereiche aller aktivierten Circuits mit einer Default-Route zusammengefasst werden, gibt es zu diesen Lückenzeiten keine Default-Route. Damit ist dann das Surfen im Internet zu diesen Zeiten ausgeschlossen!

Und noch eine letzte Bemerkung: Feiertage werden wie Sonntage behandelt.

**CIRC\_x\_CHARGEINT** Mit dieser Variable wird der Zeittakt in Sekunden angegeben. Dieser wird dann für die Kosten-Berechnung verwendet.

Die meisten Provider rechnen minutengenau ab. In diesem Fall ist der Wert '60' richtig. Bei Providern mit sekundengenaue Abrechnung setzt man die Variable entsprechend auf '1'.

Diese Variable ist nur möglich bzw. sinnvoll bei Circuits, die tatsächlich Kosten verursachen. Bei Circuits, die keine Einwahl im herkömmlichen Sinne durchführen (route, dhcp), ist diese Einstellung nicht möglich.

Standard-Einstellung: CIRC\_x\_CHARGEINT='0'

Beispiel: CIRC\_x\_CHARGEINT='60'

**CIRC\_x\_HUP\_TIMEOUT** Hier kann die Zeit in Sekunden angegeben werden, nach welcher die Verbindung beendet werden soll, wenn kein Datenverkehr mehr über den Circuit läuft. Dabei steht ein Timeout von '0' für "kein Timeout", d. h. der Router legt nicht auf und wählt sich nach einem Zwangsauflegen auch sofort wieder neu ein.

Momentan wird ein Hangup-Timeout > 0 nur für ppp-Circuits unterstützt.

Diese Eigenschaft unterscheidet generell zwischen "normalen" Circuits, die sich bei der `fli4lctrl dial`-Operation sofort einwählen (Hangup-Timeout gleich null), und "dial-on-demand"-Circuits, die nach der `fli4lctrl dial`-Operation nur bereit für eine (kommende) Einwahl sind (Hangup-Timeout größer null). Mehr Informationen hierzu finden sich in den Abschnitten "[Circuit-Zustände](#)" (Seite 89) und "[Das Programm fli4lctrl](#)" (Seite 92).

Standard-Einstellung: CIRC\_x\_HUP\_TIMEOUT='0'

Beispiel: CIRC\_x\_HUP\_TIMEOUT='600'

**CIRC\_x\_USEPEERDNS** Hiermit wird festgelegt, ob die vom Provider bei der Einwahl übergebenen DNS-Namensserver für die Dauer der Verbindung in die Konfigurationsdatei des lokalen DNS-Servers (dnsmasq) eingetragen werden sollen.

Sinnvoll ist die Nutzung dieser Option also nur bei Circuits, die entsprechende Informationen liefern. Dies betrifft i. d. R. Internet-Anbindungen via PPP und DHCP-Circuits.

Diese Option bietet den Vorteil, immer mit den am nächsten liegenden DNS-Namensservern arbeiten zu können, sofern der Provider die korrekten IP-Adressen übermittelt – dadurch geht die Namensauflösung schneller.

### 3. Basiskonfiguration

Im Falle eines Ausfalls eines DNS-Servers beim Provider werden in der Regel die übergebenen DNS-Server-Adressen sehr schnell vom Provider korrigiert.

Trotz allem ist vor jeder ersten Einwahl die Angabe eines gültigen Namensservers in `DNS_FORWARDERS` zwingend erforderlich, da sonst die erste Anfrage nicht korrekt aufgelöst werden kann. Außerdem wird beim Beenden der Verbindung die originale Konfiguration des lokalen Namensservers wieder hergestellt.

Standard-Einstellung: `CIRC_x_USEPEERDNS='no'`

Beispiel: `CIRC_x_USEPEERDNS='yes'`

**CIRC\_x\_WAIT** In der Regel wird das Wählen eines Circuits im Hintergrund durchgeführt. Will man jedoch beim Booten sicherstellen, dass sich ein Circuit erfolgreich eingewählt hat, kann man diese Variable auf die Anzahl der maximal zu wartenden Sekunden setzen. Bei '0' wird nicht gewartet.

Ein Wert größer null kann bei ppp-Circuits (d. h. wenn `CIRC_x_TYPE='ppp'` gesetzt ist) nur verwendet werden, wenn es sich um keinen Dial-on-demand-Circuit handelt, wenn also `CIRC_x_HUP_TIMEOUT='0'` gesetzt ist (bzw. die Variable gar nicht definiert wird). Der Grund hierfür ist, dass ein Warten auf einen Dial-on-demand-Circuit wenig Sinn hat, weil der Wählvorgang erst bei entsprechender Netzwerkaktivität erfolgt.

Standard-Einstellung: `CIRC_x_WAIT='0'`

Beispiel: `CIRC_x_WAIT='15'`

**CIRC\_x\_DEBUG** Mit dieser Variable können zusätzliche Debug-Ausgaben eingeschaltet werden. Dies ist Circuit-spezifisch und hat nicht zwangsläufig bei jedem Circuit-Typ sichtbare Auswirkungen.

Standard-Einstellung: `CIRC_x_DEBUG='no'`

Beispiel: `CIRC_x_DEBUG='yes'`

**CIRC\_x\_DEPS** Mit dieser Variable können Abhängigkeiten zwischen Circuits spezifiziert werden. Ein Circuit A, der von einem Circuit B abhängig ist, kann nur dann online gehen, wenn Circuit B ebenfalls online ist. Dies ist insbesondere dann nützlich, wenn Circuit A über seine Netzanbindung Infrastruktur zur Verfügung stellt, die Circuit B benötigt, etwa wenn Circuit A eine IPv4-Internetanbindung herstellt und Circuit B einen 6in4-Tunnel aufbaut.

Standard-Einstellung: `CIRC_x_DEPS=''`

Beispiel 1: `CIRC_x_DEPS='internet'`

Gelegentlich muss nicht der *gesamte* Circuit mit all seinen konfigurierten Layer-3-Protokollen online sein, damit eine Abhängigkeit erfüllt ist, sondern nur ein bestimmtes Protokoll, z.B. IPv4 oder IPv6. In diesem Fall kann man das Protokoll hinter dem Circuit angeben, mit einem Schrägstrich abgetrennt. Das folgende Beispiel zeigt eine Abhängigkeit zu einem Tag oder Circuit namens "internet", wobei es ausreicht, wenn dessen IPv4-Anbindung online ist. Dies ist z.B. für 6in4-Tunnel völlig ausreichend, denn die IPv6-Konnektivität spielt bei 6in4 naturgemäß keine Rolle (schließlich stellt ein 6in4-Tunnel gerade eine IPv6-Anbindung über eine IPv4-Anbindung her).

Beispiel 2: `CIRC_x_DEPS='internet/ipv4'`

**CIRC\_x\_CLASS\_y** Mit diesen Variablen können einem Circuit **Klassen** (Seite 92) zugeordnet werden. Durch Klassen sich Circuits logisch gruppieren, es wird somit eine Abstraktion geschaffen, die auf vielfältige Weise ausgenutzt werden kann. Eine mögliche Anwendung ist die Nutzung innerhalb von Abhängigkeiten zwischen Circuits (**CIRC\_x\_DEPS**, siehe oben), wenn mehrere Circuits eine Abhängigkeit erfüllen können.

Jede hier angegebene Klasse muss via **CIRC\_CLASS\_x\_NAME** (Seite 92) definiert werden. Ist dies nicht der Fall, wird eine Fehlermeldung ausgegeben.

Standard-Einstellung: **CIRC\_x\_CLASS\_N**='0'

Beispiel:

```
CIRC_1_NAME='DHCP-LAN'
CIRC_1_TYPE='dhcp'
CIRC_1_ENABLED='yes'
CIRC_1_DHCP_DEV='eth0'
CIRC_1_NETS_IPV4_N='1'
CIRC_1_NETS_IPV4_1='0.0.0.0/0'
CIRC_1_CLASS_N='1'
CIRC_1_CLASS_1='internet-v4'

CIRC_2_NAME='DSL-Telekom'
CIRC_2_TYPE='ppp'
CIRC_2_ENABLED='yes'
CIRC_2_PPP_TYPE='ethernet'
CIRC_2_PPP_USERID='anonymer'
CIRC_2_PPP_PASSWORD='surfer'
CIRC_2_PPP_ETHERNET_TYPE='kernel'
CIRC_2_PPP_ETHERNET_DEV='eth1'
CIRC_2_NETS_IPV4_N='1'
CIRC_2_NETS_IPV4_1='0.0.0.0/0'
CIRC_2_CLASS_N='1'
CIRC_2_CLASS_1='internet-v4'

CIRC_3_NAME='IPv6-Tunnel'
CIRC_3_TYPE='tun6in4-he'
CIRC_3_ENABLED='yes'
CIRC_3_NETS_IPV6_N='1'
CIRC_3_NETS_IPV6_1='::/0'
CIRC_3_CLASS_N='1'
CIRC_3_CLASS_1='internet-v6'
CIRC_3_DEPS='internet-v4'
```

In diesem Beispiel ist ein 6in4-HE-Tunnel (siehe Paket `ipv6`) abhängig von einem Circuit der Klasse “internet-v4”. Ob das zur Laufzeit dann die DSL- (siehe Paket `dsl`) oder DHCP-Anbindung (siehe Paket `dhcp_client`) ist, ist egal – sobald einer der beiden Circuits online ist, kann der Tunnel ebenfalls online gehen.

**CIRC\_x\_BUNDLE** Ist **CIRC\_x\_BUNDLE** nicht leer und referenziert es einen anderen gültigen Circuit, dann wird der referenzierende Circuit Teil eines so genannten “Bündels”. Gebündelte Circuits bilden zusammen *eine* logische Verbindung. Dies wird momentan nur vom Paket `ppp` unterstützt, siehe Abschnitt “Multilink PPP” (Seite ??).



Beispiel: CIRC\_x\_BUNDLE='internet-mp'

### 3.17.2. Circuit-Zustände

Jeder Circuit hat, während der Router läuft, einen der folgenden Zustände:

Zustand	Interner Name	Beschreibung
<i>inaktiv</i>	inactive	Ein Circuit ist <i>inaktiv</i> , wenn er nicht zur Einwahl herangezogen werden kann.
<i>aktiv</i>	active	Ein Circuit ist <i>aktiv</i> , wenn er zur Einwahl herangezogen werden kann.
<i>bereit</i>	ready	Ein Circuit ist <i>bereit</i> , wenn er sich bei Netzwerk-Aktivität automatisch einwählt (“dial-on-demand”).
<i>online</i>	online	Ein Circuit ist <i>online</i> , wenn die Verbindung erfolgreich aufgebaut werden konnte und die Netzwerk-Anbindung erfolgt ist.
<i>ausgefallen</i>	failed	Ein Circuit ist <i>ausgefallen</i> , wenn festgestellt wurde, dass die Verbindung über diesen Circuit nicht funktioniert. Dieser Zustand entspricht fast komplett dem Zustand <i>inaktiv</i> , mit dem einzigen Unterschied, dass ein ausgefallener Zustand von <i>circd</i> ignoriert wird (siehe hierzu den Abschnitt “Das Programm <i>circd</i> ” (Seite 95)). Dieser Zustand ist somit vor allem für einen automatisierten Fallback-Mechanismus gedacht.

Tabelle 3.11.: Circuit-Zustände

Die *Übergänge* zwischen diesen Zuständen werden teilweise mit *fli4lctrl* (Seite 92) durchgeführt, teilweise erfolgen sie automatisch. Ihre Bedeutung ist wie folgt:

Zustandsübergang	Beschreibung
<i>inaktiv</i> → <i>aktiv</i>	Ein Circuit wird aktiviert und kann sich je nach Wählmodus manuell oder automatisch einwählen. Zu diesem Zeitpunkt können noch keine Daten über den Circuit transportiert werden. Dieser Zustandsübergang kann in allen Wählmodi erfolgen. Er wird durch <i>fli4lctrl up</i> ausgelöst.
<i>aktiv</i> → <i>bereit</i>	Ein <i>aktiver</i> Circuit wird in den Zustand <i>bereit</i> versetzt, in dem eine Einwahl auf Grund von Netzwerk-Aktivität möglich ist. Zu diesem Zeitpunkt können noch keine Daten über den Circuit transportiert werden. In der Regel werden bei diesem Zustandsübergang Hintergrundprozesse gestartet, die für die folgenden Zustandsübergänge verantwortlich sind. Im Wählmodus <i>auto</i> erfolgt dieser Zustandsübergang direkt nach dem Übergang <i>inaktiv</i> → <i>aktiv</i> oder <i>bereit</i> → <i>aktiv</i> . Im Wählmodus <i>manual</i> muss dieser Zustandsübergang explizit durch <i>fli4lctrl dial</i> ausgelöst werden. Im Wählmodus <i>off</i> ist dieser Zustandsübergang nicht möglich.

Zustandsübergang	Beschreibung
<i>bereit</i> → <i>online</i>	Über den Circuit im Zustand <i>bereit</i> findet eine Einwahl statt. Nach deren erfolgreichem Abschluss können Daten über den Circuit transportiert werden, sofern für den Circuit entsprechende zu routende Netze (siehe <a href="#">CIRC_x_NETS_IPV4_y</a> (Seite 83) und <a href="#">CIRC_x_NETS_IPV6_y</a> (Seite 83)) konfiguriert sind. Je nach <a href="#">Hangup-Timeout</a> (Seite 86) erfolgt dieser Zustandsübergang direkt nach dem Zustandsübergang <i>aktiv</i> → <i>bereit</i> (Hangup-Timeout = 0), oder er wird durch eine Netzwerk-Aktivität ausgelöst (Hangup-Timeout > 0).
<i>online</i> → <i>bereit</i>	Die Wählverbindung wird beendet und die Netzwerk-Anbindung abgebaut. Danach können keine Daten mehr über den Circuit transportiert werden. Je nach <a href="#">Hangup-Timeout</a> (Seite 86) wird dieser Zustandsübergang entweder explizit vom Benutzer via <code>fli4lctrl hangup</code> angefordert (Hangup-Timeout = 0), oder er erfolgt automatisch nach einer gewissen Zeitspanne der Netzwerk-Inaktivität (Hangup-Timeout > 0).
<i>bereit</i> → <i>aktiv</i>	Ein Circuit im Zustand <i>bereit</i> wird wieder in den Zustand <i>aktiv</i> versetzt. Danach ist eine automatische (Wieder-)Einwahl auf Grund von Netzwerkaktivität nicht mehr möglich. In der Regel werden bei diesem Zustandsübergang Hintergrundprozesse beendet, die beim Zustandsübergang <i>aktiv</i> → <i>bereit</i> gestartet wurden. Im Wählmodus <i>manual</i> erfolgt dieser Zustandsübergang direkt nach dem Übergang <i>online</i> → <i>bereit</i> . Im Wählmodus <i>auto</i> muss dieser Zustandsübergang explizit durch <code>fli4lctrl hangup</code> ausgelöst werden. (Auf Grund der Semantik vom Wählmodus <i>auto</i> wird sofort wieder in den Zustand <i>bereit</i> gewechselt, siehe die Beschreibung von <i>aktiv</i> → <i>bereit</i> weiter oben.)
<i>aktiv</i> → <i>inaktiv</i>	Ein Circuit wird deaktiviert und kann künftig nicht mehr zur Einwahl herangezogen werden. Dieser Zustandsübergang kann in allen Wählmodi erfolgen. Er wird durch <code>fli4lctrl down</code> ausgelöst.

Tabelle 3.12.: Circuit-Zustandsübergänge

Nicht jeder Circuit unterscheidet effektiv zwischen *bereit* und *online*. So fallen diese Konzepte z.B. bei DHCP zusammen, weil es dort nicht möglich ist, einen Hangup-Timeout > 0 zu konfigurieren.

### 3.17.3. Wählmodus (DIALMODE)

Der Wählmodus steuert, ob und auf welche Art und Weise der fli4l-Router für die Einwahl verantwortlich ist. Der Wählmodus ist zum einen eine globale Eigenschaft, die das generelle Wählverhalten des fli4l kontrolliert. Es gibt drei Varianten:

### 3. Basiskonfiguration

Wählmodus	Beschreibung
off	In diesem Modus sind alle Circuits <i>inaktiv</i> oder <i>aktiv</i> . Der fli4l wählt weder von alleine (Netzwerkaktivität, <code>circd</code> etc.) noch auf Benutzerwunsch ( <code>fli4lctrl dial</code> , WebGUI etc.).
manual	In diesem Modus können <i>aktive</i> Circuits auf explizite Benutzeranfrage (via <code>fli4lctrl dial</code> oder über die WebGUI) in den Zustand <i>online</i> wechseln. Sowohl bei <code>fli4lctrl hangup</code> als auch bei einem konfigurierten Hangup-Timeout > 0 und entsprechend langer Netzwerk-Inaktivität wird aufgelegt, der Circuit wechselt dabei wieder in den Zustand <i>aktiv</i> . In diesem Wählmodus ignoriert <code>circd</code> die Zeitspezifikationen und schaltet keine Circuits um.
auto	In diesem Modus werden <i>aktive</i> Circuits automatisch in den Zustand <i>bereit</i> versetzt. Je nach konfiguriertem Hangup-Timeout wechselt ein solcher Circuit entweder sofort (Hangup-Timeout = 0) oder erst bei Bedarf (Hangup-Timeout > 0) in den Zustand <i>online</i> , nämlich wenn Netzwerk-Aktivität verzeichnet wird. Bei einem konfigurierten Hangup-Timeout > 0 wird bei entsprechend langer Netzwerk-Inaktivität aufgelegt, der Circuit wechselt dabei wieder in den Zustand <i>bereit</i> . Der Befehl <code>fli4lctrl dial</code> steuert hierbei <i>nicht</i> direkt die Einwahl, sondern nur, ob der Circuit <i>bereit</i> oder nicht <i>bereit</i> (d. h. nur <i>aktiv</i> ) ist. Bei externen Verbindungsabbrüchen oder beim expliziten Auflegen wird danach gleich wieder eine erneute Einwahl versucht. In diesem Wählmodus schaltet <code>circd</code> je nach Zeitspezifikation die Circuits automatisch um. <i>Ausgefallene</i> Circuits werden dabei jedoch ignoriert.

Tabelle 3.13.: Verfügbare Wählmodi

Sowohl im Wählmodus *manual* als auch im Wählmodus *auto* muss ein Circuit via `fli4lctrl up` aktiviert werden, bevor er (automatisch oder manuell via `fli4lctrl dial`) in den Zustand *bereit* wechseln kann.

Zum anderen ist der Wählmodus eine lokale Eigenschaft, die pro Circuit verwaltet wird. Der *effektive* Wählmodus ist dann das Minimum beider Wählmodi, mit der Ordnung *off* < *manual* < *auto*. Damit lässt sich z. B. erreichen, dass Circuits generell automatisch (z. B. ins Internet), einige Circuits (z. B. in die Firma) aber nur manuell gewählt werden; ein schnelles Auflegen aller Circuits ist weiterhin durch das Ändern des globalen Wählmodus auf *off* einfach zu realisieren.

Der lokale Wählmodus eines jeden Circuits ist, sofern nicht explizit via `CIRC_x_DIALMODE` (Seite 83) anders konfiguriert, initial “auto”, so dass anfangs nur der globale Wählmodus eine Rolle spielt.

**DIALMODE** Mit dieser Variable wird der initiale globale Wählmodus beim Booten festgelegt.

Eine Änderung ist im Nachhinein mit Hilfe des `fli4lctrl`-Programms, der WebGUI oder dem imon-Client möglich.

Standard-Einstellung: `DIALMODE='auto'`

Beispiel: `DIALMODE='manual'`

### 3.17.4. Circuit-Klassen

Circuit-Klassen bieten eine Möglichkeit, “artverwandte” Circuits zu gruppieren. Wenn man z. B. mehrere Möglichkeiten der Anbindung des Routers ans Internet hat, so ist doch all diesen Circuits die Default-Route gemeinsam, d. h. dass `CIRC_x_NETS_IPV4_y='0.0.0.0/0'` (für IPv4) bzw. `CIRC_x_NETS_IPV6_y='::/0'` (für IPv6) gilt. Dies kann man “herausmultiplizieren” und daraus eine eigene Klasse definieren, die man z. B. sinnigerweise “Internet” nennt:

```
CIRC_CLASS_N='1'
CIRC_CLASS_1_NAME='Internet'
CIRC_CLASS_1_NETS_IPV4_N='1'           # Circuits dieser Klasse installieren die
CIRC_CLASS_1_NETS_IPV4_1='0.0.0.0/0'  # Default-Route für IPv4...
CIRC_CLASS_1_NETS_IPV6_N='1'
CIRC_CLASS_1_NETS_IPV6_1='::/0'       # ...und für IPv6
```

Ein Circuit kann dieser Klasse “Internet” mit Hilfe der Variable `CIRC_x_CLASS_y` (Seite 88) zugeordnet werden.

Neben Routen können auch Firewall-Regeln in einer Klasse sinnvoll sein. So kann z. B. eine Port-Weiterleitung zentral für alle Internet-Circuits in der Klasse “Internet” konfiguriert werden:

```
# Fortsetzung von oben
CIRC_CLASS_1_PF_PREROUTING_N='1'
CIRC_CLASS_1_PF_PREROUTING_1='tmpl:http DNAT:@web-server'
CIRC_CLASS_1_PF_FORWARD_N='1'
CIRC_CLASS_1_PF_FORWARD_1='tmpl:http @web-server ACCEPT'
```

Diese Regeln sind dann für alle Circuits gültig, die zur Klasse “Internet” gehören.

**CIRC\_CLASS\_N** Diese Variable gibt die Anzahl der konfigurierten Circuit-Klassen an.

Standard-Einstellung: `CIRC_CLASS_N='0'`

Beispiel: `CIRC_CLASS_N='2'`

**CIRC\_CLASS\_x\_NAME** Jede Circuit-Klasse hat einen Namen. Dieser Name kann aus Buchstaben, Ziffern und dem Bindestrich ('-') bestehen. Der Name muss unter allen Circuits (siehe `CIRC_x_NAME` (Seite 82)) und Circuit-Klassen eindeutig sein.

Beispiel: `CIRC_CLASS_x_NAME='Internet'`

### 3.17.5. Das Programm `fli4lctrl`

Das Programm `fli4lctrl` ist der Zugang zum Circuit-System über die Kommandozeile. Es kann verwendet werden, um die Zustandsübergänge der Circuits herbeizuführen und um den Wählmodus zu verändern. Die möglichen Befehle lauten:

Kommando	Beschreibung
<code>up &lt;Circuit&gt;</code>	Der Circuit wird vom Zustand <i>inaktiv</i> in den Zustand <i>aktiv</i> überführt. Im Wählmodus <i>auto</i> schließt sich unmittelbar ein <code>dial &lt;Circuit&gt;</code> an, s. u.

### 3. Basiskonfiguration

Kommando	Beschreibung
<b>dial</b> <Circuit>	Der Circuit wird vom Zustand <i>aktiv</i> in den Zustand <i>bereit</i> überführt. Bei einem Hangup-Timeout = 0 folgt unmittelbar ein Wählvorgang mit dem Ziel, den Circuit in den Zustand <i>online</i> zu überführen. Bei einem Hangup-Timeout > 0 wird nicht sofort gewählt, sondern auf entsprechende Netzwerk-Aktivität gewartet. Im Wählmodus <i>off</i> wird das Kommando ignoriert.
<b>autodial</b> <Circuit>	Der Circuit wird genauso wie beim Befehl <b>fli4lctrl dial</b> vom Zustand <i>aktiv</i> in den Zustand <i>bereit</i> versetzt, aber <i>nur</i> , wenn der effektive Wählmodus des Circuits <i>auto</i> ist. In allen anderen Fällen wird der Befehl ignoriert.
<b>dial</b>	Es findet die Initiierung der Einwahl auf allen aktiven Circuits statt.
<b>hangup</b> <Circuit>	Der Circuit wird von den Zuständen <i>online</i> und <i>bereit</i> in den Zustand <i>aktiv</i> überführt. Im Wählmodus <i>auto</i> erfolgt anschließend ein <b>dial</b> <Circuit>.
<b>hangup</b>	Es findet ein Auflegen aller Circuits statt, die in den Zuständen <i>bereit</i> oder <i>online</i> sind.
<b>down</b> <Circuit>	Der Circuit wird von allen anderen Zuständen in den Zustand <i>inaktiv</i> überführt, unabhängig vom Wählmodus. Falls der Circuit vorher im Zustand <i>bereit</i> oder <i>online</i> war, wird vorher ein <b>hangup</b> <Circuit> ausgeführt, s. o.
<b>fail</b> <Circuit>	Der Circuit wird von allen anderen Zuständen in den Zustand <i>ausgefallen</i> überführt, unabhängig vom Wählmodus. Falls der Circuit vorher im Zustand <i>bereit</i> oder <i>online</i> war, wird vorher ein <b>hangup</b> <Circuit> ausgeführt, s. o.

### 3. Basiskonfiguration

Kommando	Beschreibung
<code>dialmode global</code> <Wählmodus>	<p>Der angegebene Wählmodus wird global eingestellt. Pro Circuit wird der <i>effektive</i> Wählmodus als Minimum des globalen und lokalen (s.u.) Wählmodus bestimmt. Folgende Zustandsübergänge finden statt:</p> <ul style="list-style-type: none"> <li>• Ein effektiver Moduswechsel <math>* \rightarrow \text{off}</math> impliziert ein <b>hangup</b> auf allen Circuits in den Zuständen <i>bereit</i> oder <i>online</i> (dies entspricht somit einem <code>fli4lctrl hangup</code>).</li> <li>• Ein effektiver Moduswechsel <math>* \rightarrow \text{manual}</math> impliziert ein <b>hangup</b> auf allen Circuits im Zustand <i>bereit</i>. Circuits im Zustand <i>online</i> legen <i>nicht</i> auf; hier greift der neue Wählmodus erst <i>nach</i> dem nächsten Auflegen.</li> <li>• Ein effektiver Moduswechsel <math>* \rightarrow \text{auto}</math> impliziert ein <b>dial</b> auf allen <i>aktiven</i> Circuits (dies entspricht somit einem <code>fli4lctrl dial</code>).</li> </ul>
<code>dialmode local</code> <Circuit> <Wählmodus>	Der angegebene Wählmodus wird lokal für den Circuit eingestellt. Für die sich daraus ergebenden Zustandsänderungen siehe oben.

Tabelle 3.14.: fli4lctrl-Befehle

Kommando	Beschreibung
<code>status</code> <Circuit>	Es wird der aktuelle Zustand des zugehörigen Circuits ausgegeben.
<code>status</code>	Der aktuelle Online-Status des Routers wird ausgegeben. Der Rückgabecode ist 0, falls der Router <i>online</i> ist, und 1, falls er <i>offline</i> ist oder ein Fehler aufgetreten ist. Wann genau der Router <i>online</i> ist, wird im Abschnitt <a href="#">Wann ist mein Router online?</a> (Seite 96) erläutert.
<code>dialmode global</code>	Der aktuelle globale Wählmodus wird zurückgegeben.
<code>dialmode local</code> <Circuit>	Der aktuelle lokale Wählmodus für den angegebenen Circuit wird zurückgegeben.
<code>dialmode effective</code> <Circuit>	Der aktuelle effektive Wählmodus für den angegebenen Circuit wird zurückgegeben.
<code>list states</code>	Eine Liste aller Circuits mit den zugehörigen Zuständen wird ausgegeben.
<code>list dialmodes</code>	Eine Liste aller Circuits mit den zugehörigen lokalen und effektiven Wählmodi wird ausgegeben.

Kommando	Beschreibung
<code>list classes</code>	Eine Liste aller Circuits mit den zugehörigen Klassen wird ausgegeben.
<code>list deps</code>	Eine Liste aller Circuits mit den zugehörigen Abhängigkeiten wird ausgegeben. Erfüllte Abhängigkeiten werden entsprechend markiert.
<code>show</code>	Ein Alias für <code>list states</code> .

Tabelle 3.15.: `fli4lctrl`-Statusbefehle

Zu beachten ist, dass statt des Identifikators eines Circuits (z. B. “`circ1`”) alternativ auch

- sein Alias (z. B. “`ppp0`”),
- sein Name (z. B. “T-Com DSL”) oder
- ein aktives, ihm zugeordnetes Schlagwort (z. B. “`internet-v4`”)

verwendet werden kann.

### 3.17.6. Das Programm `circd`

Der `circd` ist ein Dämon, der anhand von Zeit-Spezifikationen Circuits automatisch im Hintergrund umschaltet. Dabei werden neben der Zeit- und Kosten-Spezifikation (`CIRC_x_TIMES`) auch die Priorität eines Circuits (`CIRC_x_PRIORITY`) sowie dessen Zustand (*ausgefallen* oder nicht) berücksichtigt.

Für einen beliebigen Zeitpunkt funktioniert das Auswählen wie folgt:

1. Zuerst werden alle Circuits gesucht, die für den betreffenden Zeitpunkt aktiviert sind (Typ ‘Y’).
2. Aus dieser Menge werden zuerst all jene betrachtet, die zur höchsten verwendeten Prioritätsklasse gehören (deren `CIRC_x_PRIORITY`-Wert also am niedrigsten ist).
3. Aus dieser Menge werden alle Circuits entfernt, die nicht nutzbar sind, die also im Zustand *ausgefallen* sind.
4. Wenn die resultierende Menge nicht leer ist, ist der Algorithmus beendet, und die resultierenden Circuits werden ausgewählt. Wenn die resultierende Menge leer ist, werden die Schritte 2 und 3 für die nächst niedrigere Prioritätsklasse wiederholt.
5. Falls alle via ‘Y’ aktivierten Circuits nicht nutzbar sind, werden alle Circuits gesucht, die für den betreffenden Zeitpunkt als LCR-Circuits vorgemerkt sind (Typ ‘L’).
6. Aus dieser Menge werden zuerst all jene betrachtet, die zur höchsten verwendeten Prioritätsklasse gehören (deren `CIRC_x_PRIORITY`-Wert also am niedrigsten ist).
7. Aus dieser Menge werden alle Circuits entfernt, die nicht nutzbar sind, die also im Zustand *ausgefallen* sind.
8. Aus dieser Menge werden diejenigen Circuits bestimmt, die am wenigsten kosten.

9. Wenn die resultierende Menge nicht leer ist, ist der Algorithmus beendet, und die resultierenden Circuits werden ausgewählt. Wenn die resultierende Menge leer ist, werden die Schritte 6 bis 8 für die nächst niedrigere Prioritätsklasse wiederholt.
10. Falls danach immer noch keine Circuits übrig bleiben, wird kein Circuit ausgewählt.

Einige Anmerkungen zum Algorithmus:

- Mit ‘Y’ aktivierte Circuits “drängeln” sich immer vor alle mit ‘L’ aktivierten Circuits. Damit ist der Nutzer in der Lage, für bestimmte Zeiträume die Least-cost-Funktionalität zu deaktivieren und die Menge der zu verwendenden Circuits explizit einzustellen. Dies ist immer dann nützlich, wenn man aus irgendwelchen Gründen auf einen bestimmten Circuit angewiesen ist, etwa weil man die Dienste des Providers in Anspruch nehmen will, auf die man nur dann Zugriff hat, wenn man sich über den Zugang des Providers einwählt.
- Bei mit ‘Y’ aktivierten Circuits werden die Kosten bei der Auswahl ignoriert, es werden somit immer *alle* Circuits derselben Prioritätsklasse *gleichzeitig* ausgewählt. Das ist konsistent mit der Bedeutung des Typs ‘Y’ (aktiviere den Circuit unabhängig von den Kosten). Will man teurere Circuits nicht mit günstigeren gleichzeitig aktivieren, muss man den teureren Circuits eine niedrigere Priorität zuordnen.
- Die Priorität wird *vor* den Kosten berücksichtigt. Damit ist der Nutzer in der Lage, auch im Least-cost-Betrieb unabhängig von den tatsächlichen Kosten Circuits zu ordnen, da die Priorität eine benutzerdefinierte und nach Belieben veränderbare Einstellung ist, die Kosten jedoch vom Provider vorgegeben sind und für eine korrekte Abrechnung keine Veränderungen zulässig sind. Circuits zuerst nach Priorität zu ordnen ist insbesondere dann vorteilhaft, wenn ein Circuit zwar günstiger ist als andere (oder gar umsonst ist), aber dafür die Datenübertragungsrate sehr langsam ist. In diesem Fall möchte man den langsamen Circuit aller Voraussicht nach nur als Fallback nutzen, also falls alle schnelleren (und teureren) Circuits ausfallen. Bei einer umgekehrten Reihenfolge (erst Kosten, dann Priorität berücksichtigen) würde immer der kostenlose und langsame Circuit ausgewählt, ohne dass man mit der Priorität irgendetwas daran verändern könnte.
- Der Algorithmus kann durchaus mehrere Circuits als Ergebnis liefern. Dies ist unproblematisch, falls die von den Circuits gerouteten Netzwerke sich nicht überlappen. Überlappenden Netzwerke führen zu einem Fehler, und es wird nur ein Circuit aktiviert (siehe hierzu die Beschreibung der Variablen `CIRC_x_UP`); eine automatische Verteilung des Datenverkehrs auf mehrere Circuits (Stichwort “Load Balancing”) ist momentan noch nicht möglich.

#### 3.17.7. Wann ist mein Router online?

Was auf den ersten Blick trivial erscheint, ist auf den zweiten Blick eine recht knifflige Frage: Wann ist ein fli4l-Router “online”? Ist nur ein Circuit definiert, ist die Frage relativ einfach zu beantworten: Der Router ist genau dann *online*, wenn der Circuit im Zustand *online* ist. Bei mehreren Circuits, die verschiedene Netze routen, sich teilweise gegenseitig ausschließen und auch u. U. keine Wählverbindung eingehen ist die Frage schon schwieriger zu beantworten. Auch einige auf der Hand liegende Ansätze liefern nicht immer die gewünschte Antwort:



### 3. Basiskonfiguration

1. *Der Router ist online, wenn mindestens ein Circuit online ist.* Diese Regelung ist ungünstig, weil bereits die Existenz einer einzigen Route den Router in den Zustand *online* versetzt, da Routen intern über Circuits abgewickelt werden. Und ein Router ohne Routen ist nicht sehr interessant. . .
2. *Der Router ist online, wenn alle Circuits online sind.* Diese Regelung ist immer dann falsch, wenn sich Circuits gegenseitig ausschließen, wenn also beispielsweise zwei DSL-Circuits konfiguriert werden, die beide eine Internet-Anbindung herstellen und deshalb nicht parallel aktiviert werden können.
3. *Der Router ist online, wenn alle Dialup-Circuits online sind.* Diese Regelung ist besser als Regelung 1, weil sie Route-Circuits ausschließt, aber auch sie löst nicht das Problem der sich ausschließenden Circuits.
4. *Der Router ist online, wenn mindestens ein Dialup-Circuit online ist.* Dies versetzt den Router in den Zustand *online*, wenn z. B. zwar eine Wählverbindung in die Firma besteht, aber die Internet-Verbindung inaktiv ist. Das kann gewünscht sein, muss aber nicht.
5. *Der Router ist online, wenn ein Circuit mit Default-Route online ist.* Diese Regelung entspricht dem Zustand vor fl4l 4.0. Dies versetzt den Router in den Zustand *offline*, wenn eine Wählverbindung in die Firma besteht, aber die Internet-Verbindung inaktiv ist. Das kann gewünscht sein, muss aber nicht.
6. *Der Router ist online, wenn für jedes konfigurierte, zu routende Netz ein Circuit online ist.* Auch diese Regelung versetzt den Router in den Zustand *offline*, wenn eine Wählverbindung in die Firma besteht, aber die Internet-Verbindung inaktiv ist. Das kann gewünscht sein, muss aber nicht.

Anstatt nun eine feste Regel vorzugeben und im Zweifelsfall genau das zu tun, was der Benutzer *nicht* erwartet, wurde ein zweistufiges Verfahren gewählt. Der Standard-Fall wählt die rückwärtskompatible Regelung 5. Alternativ kann über die Variable `CIRC_ONLINE` eine Menge von Circuits (oder Circuit-Schlagwörtern) spezifiziert werden, die für die Online-Bewertung berücksichtigt werden sollen. Sollen also sowohl Internet- als auch Firma-Anbindungen berücksichtigt werden, und werden die entsprechenden Circuits mit den Schlagwörtern “internet” und “firma” gekennzeichnet, so ist bei `CIRC_ONLINE='internet firma'` der Router *online*, wenn der “internet”- oder der “firma”-Circuit (oder beide) online sind; bei `CIRC_ONLINE='internet'` hat der Router den Zustand *offline*, wenn keine Internet-Verbindung aktiv ist, unabhängig davon, ob eine Verbindung zur Firma besteht oder nicht.

**CIRC\_ONLINE** Diese Variable beinhaltet eine Liste von Circuits (oder Schlagwörtern), die bei der Bestimmung des Online-Zustands des Routers berücksichtigt werden. Ist die Variable vorhanden und die Liste nicht leer, ist der Router genau dann *online*, wenn mindestens ein Circuit aus der Liste *online* ist bzw. wenn mindestens ein Schlagwort aus der Liste aktiv ist. Ist die Liste leer oder die Variable nicht vorhanden, greift die rückwärtskompatible Regelung, nach welcher der Router *online* ist, wenn die Default-Route aktiv ist.

Standard-Einstellung: `CIRC_ONLINE=''`

### 3.17.8. Sonstige Einstellungen

**IP\_DYN\_ADDR** Wird eine Verbindung mit dynamischer Adressvergabe verwendet, ist `IP_DYN_ADDR` auf 'yes' zu stellen, ansonsten auf 'no'. Die meisten Internet-Provider verwenden eine dynamische Adressvergabe.

Standard-Einstellung: `IP_DYN_ADDR='yes'`

**OPT\_CIRCUIT\_STATUS** Diese Variable aktiviert einen Circuit-Status-Monitor auf der dritten fli4l-Console. Diese lässt sich mit der Tastenkombination `Alt+F3` aktivieren, ein Wechseln zurück zur Login-Konsole wird mit Hilfe der Tastenkombination `Alt+F1` bewerkstelligt. Der Monitor zeigt jeden Circuit mit Namen, Typ, Alias, Schnittstelle und Status an.

Standard-Einstellung: `OPT_CIRCUIT_STATUS='no'`

Beispiel: `OPT_CIRCUIT_STATUS='yes'`

## 3.18. Spezielle Circuit-Typen im base-Paket

### 3.18.1. Circuits vom Typ "route"

Ein Circuit vom Typ "route" dient dazu, eine vorkonfigurierte *Route* zu verwalten. Das Einwählen entspricht dem Aufbau, der Route, das Auflegen dem Abbau. Ein einzelner route-Circuit kann das Routen mehrerer IPv4- und IPv6-Netze (letzteres nur bei aktivierter IPv6-Unterstützung) gleichzeitig steuern. Insofern ist ein route-Circuit flexibler als die Konfiguration einer Route via `IP_ROUTE_x`, weil dort immer nur ein Netz pro Route angegeben werden kann.

Neben den allgemeinen Circuit-Variablen, die in Abschnitt "[Circuits allgemein](#)" (Seite 81) beschrieben sind, sind zusätzlich die folgenden Angaben erforderlich:

**CIRC\_x\_ROUTE\_DEV** In dieser Variablen ist die Netzwerk-Schnittstelle vermerkt, über welche die Route geht. Diese Angabe ist optional, sofern mindestens eine Gateway-Angabe (siehe unten) vorliegt und daraus die Netzwerk-Schnittstelle abgeleitet werden kann. Es können Schnittstellen-Namen ("`eth0`"), Schnittstellen-Referenzen ("`IP_NET_1_DEV`", "`IPV6_NET_1_DEV`"), Circuits ("`ppp0`") und Schlagwörter ("`internet-v4`") genutzt werden, wobei Routen über dynamische, von Circuits kontrollierte Schnittstellen eher esoterischer Natur sind und nur dann verwendet werden sollten, wenn man sich über die auftretenden dynamischen Effekte im Klaren ist.

Wenn diese Variable leer ist und beide Gateway-Adressen (IPv4 und IPv6) gesetzt sind (s. u.), müssen diese derselben Schnittstelle zugeordnet sein. Das liegt daran, dass jedem Circuit genau eine Netzwerkschnittstelle zugeordnet sein muss (und nicht mehrere).

Beispiel 1: `CIRC_x_ROUTE_DEV='IP_NET_2_DEV'`

Beispiel 2: `CIRC_x_ROUTE_DEV='pppoe-v6'`

**CIRC\_x\_ROUTE\_GATEWAY\_IPV4** Diese Variable enthält die Adresse des Next-Hop-Routers (gemeinhin "Gateway" genannt) für die zu routenden IPv4-Netze. Sie kann nur dann leer sein, wenn `CIRC_x_ROUTE_DEV` gesetzt ist; in diesem Fall wird eine Route ohne Gateway generiert. Die Gateway-Adresse kann mit einem Circuit kombiniert werden, wobei Routen über dynamische, von Circuits kontrollierte Gateways eher esoterischer Natur sind und

### 3. Basiskonfiguration

nur dann verwendet werden sollten, wenn man sich über die auftretenden dynamischen Effekte im Klaren ist.

Beispiel 1: `CIRC_x_ROUTE_GATEWAY_IPV4='192.168.1.254'`

Beispiel 2 (esoterisch): `CIRC_x_ROUTE_GATEWAY_IPV4='{dhcp}0.0.0.253'`

Im letzten Beispiel wird die Gateway-Adresse dynamisch aus dem per DHCP zugewiesenen Netz (z. B. 192.168.12.0/24) und dem angegebenen Host-Anteil gebildet (in diesem Fall also 192.168.12.253). Dies ist nur dann sinnvoll, wenn man weiß, dass egal welches Netzpräfix dem fli4l-Router von dem DHCP-Server zugewiesen wird, ein bestimmter Router an der Host-Adresse 253 zu finden ist. Normalerweise sollte der via DHCP übermittelte Router alle Routen behandeln können, somit sollte man die zu routenden Netze in einem solchen Fall direkt im jeweiligen DHCP-Circuit vermerken und nicht einen route-Circuit dafür verwenden.

**CIRC\_x\_ROUTE\_GATEWAY\_IPV6** Diese Variable enthält die Adresse des Next-Hop-Routers (gemeinhin “Gateway” genannt) für die zu routenden IPv6-Netze. Es gilt all das für `CIRC_x_ROUTE_GATEWAY_IPV4` Gesagte. Weiterhin kann diese Variable nur bei aktivierter IPv6-unterstützung genutzt werden.

Beispiel 1: `CIRC_x_ROUTE_GATEWAY_IPV6='2001:db8:42::1'`

Beispiel 2 (esoterisch): `CIRC_x_ROUTE_GATEWAY_IPV6='{dhcipv6}::0:0:0:254'`

## 4. Pakete

Neben der Basisinstallation (BASE) gibt es weitere Pakete. Diese enthalten ein oder mehrere “OPTs”<sup>1</sup>, die bei Bedarf zu BASE hinzuiinstalliert werden können. Einige dieser Pakete sind Bestandteil des Basis-Paketes, andere kommen separat. Eine Übersicht über die vom fli4l-Team bereitgestellten Pakete finden Sie auf der Download-Seite (<http://www.fli4l.de/download/stabile-version/>), die von anderen Autoren bereitgestellten Pakete sind in der OPT-Datenbank ([http://extern.fli4l.de/fli4l\\_opt-db3/](http://extern.fli4l.de/fli4l_opt-db3/)) zu finden. Im folgenden werden die vom fli4l-Team bereitgestellten Pakete beschrieben.

### 4.1. Werkzeuge im Basispaket

Im Basispaket befinden sich die folgenden OPTs:

Name	Beschreibung
OPT_SYSLOGD	<a href="#">Werkzeug zum Protokollieren von Systemmeldungen</a> (Seite 100)
OPT_KLOGD	<a href="#">Werkzeug zum Protokollieren von Kernmeldungen</a> (Seite 102)
OPT_LOGIP	<a href="#">Werkzeug zum Protokollieren von WAN-IP-Adressen</a> (Seite 102)
OPT_Y2K	<a href="#">Datumskorrektur bei nicht Y2K-festen Rechnern</a> (Seite 102)
OPT_PNP	<a href="#">Installation der isapnp-Werkzeuge</a> (Seite 103)
OPT_HOTPLUG_PCI	<a href="#">Aktivierung von PCI-Hotplugging</a> (Seite 105)

#### 4.1.1. OPT\_SYSLOGD – Protokollieren von Systemmeldungen

Viele Programme verwenden die Syslog-Schnittstelle, um Meldungen auszugeben. Damit diese auch auf der Konsole sichtbar werden, muss in diesem Falle der Daemon syslogd gestartet werden.

Sind Debug-Meldungen gewünscht, stellt man OPT\_SYSLOGD auf ‘yes’, ansonsten auf ‘no’.

Siehe auch ISDN\_CIRC\_x\_DEBUG (Seite ??) und PPPOE\_DEBUG (Seite ??).

Standard-Einstellung: OPT\_SYSLOGD=‘no’

**SYSLOGD\_RECEIVER** Mit SYSLOGD\_RECEIVER kann man festlegen, ob fli4l Syslog-Nachrichten vom Netzwerk empfangen soll oder nicht.

**SYSLOGD\_DEST\_N SYSLOGD\_DEST\_x** Mit SYSLOGD\_DEST\_x gibt man Ziele an, wohin die System-Meldungen, die von syslogd entgegengenommen werden, ausgegeben werden. Im Normalfall ist dies die Konsole von fli4l, also

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

Möchte man eine Datei als Ziel verwenden, ist z.B. einzutragen:

---

<sup>1</sup>Abk. für “OPTionales Modul”

#### 4. Pakete

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

Ist ein sog. Loghost im Netz vorhanden, können die Meldungen auch auf diesen Rechner umgeleitet werden – unter Angabe der IP-Adresse.

Beispiel:

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

Das @-Zeichen ist dann der IP-Adresse voranzustellen.

Wenn die Systemmeldungen an mehrere Ziele “ausgeliefert” werden sollen, ist es nötig, die Variable `SYSLOGD_DEST_N` (Anzahl der Ziele) entsprechend zu erhöhen und die Variablen `SYSLOG_DEST_1`, `SYSLOG_DEST_2` usw. zu füllen.

Der Parameter “\*. \*” bedeutet, dass sämtliche Meldungen protokolliert werden. Man kann jedoch die Meldungen für bestimmte Ziele über sog. “Prioritäten” einschränken. In diesem Fall ersetzt man das Sternchen (\*) hinter dem Punkt (.) durch eines der folgenden Schlüsselwörter:

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

Die Reihenfolge in der Liste spiegelt dabei das “Gewicht” der Meldungen wider. Die Schlüsselwörter “error”, “warn” und “panic” sind veraltet und sollten nicht mehr verwendet werden.

Vor dem Punkt kann eine sog. “Facility” statt des Sternchens (\*) eingetragen werden. Eine Erklärung würde aber hier zu weit gehen. Der geneigte Leser kann hierfür eine Suchmaschine seiner Wahl bemühen. Eine Übersicht über mögliche Facilities finden sich auf den Manpages der `syslog.conf`:

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

Normalerweise ist das Sternchen aber vollkommen ausreichend. Beispiel:

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

Nicht nur Unix-/Linux-Rechner können als Loghost dienen, sondern auch Windows-Rechner. Auf <http://www.fli4l.de/sonstiges/links/> findet man Verweise auf entsprechende Software. Die Verwendung eines Loghosts wird dringend empfohlen, wenn eine detaillierte Protokollierung gewünscht ist. Auch hilft die Protokollierung bei der Fehlersuche. Auch imonc “versteht” als Windows-Client das Syslog-Protokoll und kann die Meldungen in einem Fenster ausgeben.

Leider lassen sich die Boot-Meldungen von `fli4l` nicht über `syslogd` umlenken. Jedoch kann man `fli4l` auch so konfigurieren, dass die Konsole ein serielles Terminal (bzw. Terminalemulation) ist. Wie das geht, steht in dem Abschnitt [Konsolen-Einstellungen](#) (Seite 30).

**SYSLOGD\_ROTATE** Mit `SYSLOGD_ROTATE` kann man festlegen, ob `fli4l` Syslog-Nachrichten einmal täglich rotiert. Es werden dabei die Meldungen der letzten `x` Tage archiviert.

**SYSLOGD\_ROTATE\_DIR** Mit der optionalen Variable `SYSLOGD_ROTATE_DIR` kann man festlegen, dass die rotierten Syslog-Dateien nicht in `/var/log/` sondern in das angegebene Verzeichnis rotiert werden.

**SYSLOGD\_ROTATE\_MAX** Mit der optionalen Variablen `SYSLOGD_ROTATE_MAX` legt man die Anzahl der archivierten/rotierten Syslog-Dateien fest.

**SYSLOGD\_ROTATE\_AT\_SHUTDOWN** Mit der optionalen Variable `SYSLOGD_ROTATE_AT_SHUTDOWN` kann man den das Rotieren beim Herunterfahren des Routers deaktivieren. Dies sollte man jedoch nur einstellen, wenn die syslog-Dateien bereits auf ein nichtflüchtiges Ziel geschrieben werden.

### 4.1.2. OPT\_KLOGD – Protokollieren von Kernelmeldungen

Viele der auftretenden Fehler – z.B. fehlgeschlagene Einwahl – werden vom Linux-Kernel direkt auf die Konsole geschrieben. Mit `OPT_KLOGD='yes'` werden diese Meldungen an den `syslogd` umgelenkt, welcher diese dann entweder in Dateien protokollieren oder an einen Loghost weiterleiten kann, s.o. Dann hat man auf der `fli4l`-Konsole (fast) Ruhe.

Empfehlung: Wenn Sie `OPT_SYSLOGD='yes'` benutzen, sollte man auch `OPT_KLOGD='yes'` setzen.  
Standard-Einstellung: `OPT_KLOGD='no'`

### 4.1.3. OPT\_LOGIP – Protokollieren von WAN-IP-Adressen

Mit `LOGIP` ist es möglich, die WAN-IP in einer Log-Datei festzuhalten. Mit Angabe von `OPT_LOGIP='yes'` wird die Funktion aktiviert.

Standard-Einstellung: `OPT_LOGIP='no'`

**LOGIP\_LOGDIR** Verzeichnis der Log-Datei festlegen

Mit `LOGIP_LOGDIR` wird das Verzeichnis festgelegt, in welchem die Log-Datei angelegt wird oder `'auto'` für autodetect.

Standard-Einstellung: `LOGIP_LOGDIR='auto'`

### 4.1.4. OPT\_Y2K – Datumskorrektur bei nicht Y2K-festen Rechnern

Meist werden `fli4l`-Router aus alten Hardware-Teilen zusammengesetzt. Dabei kann das BIOS des Mainboards nicht Y2K-fest sein. Das kann dazu führen, dass bei einer Einstellung des Datums auf den 27.05.2000 im BIOS beim nächsten Booten der 27.05.2094 im BIOS zu finden ist! Linux zeigt dann übrigens den 27.05.1994 an.

Eigentlich kann das eingestellte Datum für den `fli4l`-Router egal sein und sollte deshalb keine Rolle spielen. Wird `fli4l` jedoch als Least-Cost-Router eingesetzt, kann dies sehr wohl eine Rolle spielen.

Grund: Der 27.05.1994 war ein Freitag, der 27.05.2000 jedoch ein Samstag. Und am Wochenende gibt's günstigere Tarife bzw. günstigere Provider ...

Eine erste Lösung lautet: Das BIOS-Datum wird vom 27.05.2000 auf den 28.05.1994 gestellt. Das war ebenso ein Samstag. Damit ist das Problem aber noch nicht gänzlich gelöst, denn `fli4l`

verwendet nicht nur den Wochentag und die momentane Uhrzeit für das LC-Routing, sondern berücksichtigt auch die Feiertage.

#### **Y2K\_DAYS** – N Tage auf Systemdatum addieren

Da das gesetzte Datum genau 2191 Tage zum tatsächlichen Datum differiert, werden bei Angabe von

```
Y2K_DAYS='2191'
```

auf das BIOS-Datum 2191 Tage addiert und dieses dann als Linux-Datum gesetzt. Das BIOS-Datum bleibt davon jedoch unberührt, sonst würde beim nächsten Booten das Datum wieder das Jahr 2094 (bzw. 1994) aufweisen.

Es gibt noch eine Alternative:

Mit dem Zugriff auf einen Time-Server kann sich fli4l die aktuelle Datum/Uhrzeit aus dem Internet holen. Dafür steht das Paket CHRONY (Seite ??) zur Verfügung. Beide Einstellungen lassen sich kombinieren. Das ist sinnvoll, um mit Y2K\_DAYS zunächst das Datum schon einmal zu korrigieren und anschließend über den Time-Server die genaue Uhrzeit einzustellen.

Wer mit Y2K keine Probleme hat: Variable OPT\_Y2K='no' setzen und einfach nicht mehr darüber nachdenken ...

#### **4.1.5. OPT\_PNP – Installation von isapnp tools**

Teilweise müssen ISAPnP-Karten über das Werkzeug "isapnp" konfiguriert werden. Dies betrifft insb. ISDN-Karten mit ISDN\_TYPE 7, 12, 19, 24, 27, 28, 30 und 106 – aber nur, wenn es sich auch wirklich um ISAPnP-Karten handelt.

Zur Konfiguration ist die Erstellung einer Konfigurations-Datei etc/isapnp.conf notwendig. Hier eine Kurzanleitung zur Erstellung:

- In <config>/base.txt die Variable OPT\_PNP='yes' und MOUNT\_BOOT='rw' setzen
- fli4l booten – die ISAPnP-Karte wird wahrscheinlich nicht erkannt
- Auf fli4l-Konsole eingeben:

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

Damit ist die Konfiguration auf dem Boot-Medium gespeichert.

Weiter auf PC (Unix/Linux/Windows):

- Die Datei isapnp.conf vom Boot-Medium nach <config>/etc/isapnp.conf kopieren
- isapnp.conf mit Editor öffnen, bearbeiten und abspeichern  
Die vorgegebenen Werte kann man hier beibehalten oder auch durch andere ersetzen, die man aus den möglichen Werten wählt. Relevant sind dabei die folgenden Zeilen im folgenden Beispiel:

#### 4. Pakete

```
#      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) – Hier kann als „BASE“ eine Adresse zwischen die angegebene Minimum und Maximum eingegeben werden, wobei man das „base alignment“ in betracht ziehen muss. Bei mehr als einer ISA-Karte im System muss immer darauf geachtet werden, dass es hier keine Überschneidungen gibt, achte dabei auch auf die benötigte Anzahl Adressen (number of addresses required).

2) – Hier kann aus der angezeigten Liste ein IRQ eingesetzt werden. Dabei ist 2(9), 3, 4, 5 und 7 eher eine schlechte Wahl, da sich diese normalerweise mit den Seriellen und Parallelen Schnittstellen bzw der Cascadierung ins Gehege kommen.

ISA Karten können IRQs nicht teilen, deshalb darf ein für diese Karte verwendeter nicht anderweitig belegt sein.

- Die entsprechenden Daten (IRQ/IO) in die <config>/isdn.txt übernehmen
- Es ist nötig in der <config>/base.txt die OPT\_PNP Einstellung auf 'yes' zu belassen, anderenfalls werden die erforderliche Dateien nicht mit auf das Boot-Medium kopiert. Die Einstellung MOUNT\_BOOT kann man beliebig ändern.
- Neues Boot-Medium erzeugen

**Die automatisch generierte Datei ist im Unix-Format gespeichert und enthält keine CRs. Startet man unter Windows den Notepad-Editor, zeigt dieser alle Zeilen in einer einzigen Zeile an. Der DOS-Editor "edit" kann jedoch mit Unix-Dateien umgehen. Er speichert sie dann als DOS-Datei (mit CRs) ab.**

Abhilfe:

- DOS-Box starten
- In das Verzeichnis <config>/etc wechseln
- Eingeben: edit isapnp.conf
- Datei bearbeiten und abspeichern

Anschließend kann die Datei auch wieder mit Notepad bearbeitet werden.

Man kann auch unter Windows einfach den Wordpad-Editor verwenden.

Die zusätzlich generierten CRs werden beim Booten von fl4l wieder herausgefiltert. Sie stören also nicht.

Zunächst sollte man versuchen, ohne OPT\_PNP auszukommen. Wird die Karte nicht erkannt, sollte wie oben beschrieben vorgegangen werden.



Bei einem Update auf eine neuere fli4l-Version kann die früher erstellte Datei isapnp.conf weiter verwendet werden.

Standard-Einstellung: `OPT_PNP='no'`

##### 4.1.6. `OPT_HOTPLUG_PCI` – Aktivieren von PCI-Hotplugging

Mit `OPT_HOTPLUG_PCI='yes'` werden Module auf den fli4l kopiert und beim Booten geladen, die PCI-Hotplugging aktivieren, d.h. die Unterstützung für das Hinzufügen und Entfernen von PCI-Adaptern zur Laufzeit. Für diese Funktionalität muss ein passender PCI-Hotplug-Controller vorhanden sein.

Für das Hinzufügen und Entfernen von virtuellen Geräten in *Virtualisierungsumgebungen* wie KVM muss diese Option *nicht* aktiviert sein, da dies über ACPI-Mechanismen geschieht und der zugehörige ACPI-Treiber dauerhaft im Kernel aktiviert ist.

## 5. Erzeugen der fli4l Archive/Bootmedien

Sind alle Konfigurationsarbeiten erledigt, können die fli4l Archive/Bootmedien, sei es eine bootfähige Compact-Flash, ein bootfähiges ISO-Image oder nur die zum Remote-Update benötigten Dateien, erstellt werden.

### 5.1. Erzeugen der fli4l Archive/Bootmedien unter Linux bzw. anderen Unix-Derivaten und Mac OS X

Dies geschieht mit Hilfe von Scripts (`.sh`), die im fli4l Wurzelverzeichnis zu finden sind.

```
mkfli4l.sh
```

Das Build-Script erkennt selbständig die unterschiedlichen [Bootvarianten](#) (Seite 24). Der einfachste Aufruf sieht unter Linux so aus:

```
sh mkfli4l.sh
```

Die Aktionen des Build-Scripts werden durch drei Mechanismen gesteuert:

- Konfigurationsvariable `BOOT_TYPE` aus der `<config>/base.txt`
- Konfigurationsdatei `<config>/mkfli4l.txt`
- Parameter des Build-Scripts

An Hand der Konfigurationsvariable `BOOT_TYPE` (Seite 24) entscheidet sich, welche Aktion des Build-Scripts ausgeführt wird:

- Erstellen eines bootfähigen fli4l CD-ISO-Images
- Bereitstellen der fli4l Dateien, zwecks Remote-Update
- Erzeugen der fli4l Dateien und direktes Remote-Update per SCP
- usw.

Die Beschreibung der Variablen der Konfigurationsdatei `<config>/mkfli4l.txt` finden Sie im Kapitel [Steuerungsdatei mkfli4l.txt](#) (Seite 114).

### 5.1.1. Kommandozeilenoptionen

Der letzte Steuerungsmechanismus ist das Anhängen von Optionsparametern an den Aufruf des Build-Script auf der Kommandozeile. Die Steuerungsmöglichkeiten entsprechen denen der Steuerungsdatei `mkfli4l.txt`. Die Angabe von Optionsparametern überschreiben die Werte aus der Steuerungsdatei. Aus Komfortgründen unterscheiden sich die Namen der Optionsparameter von den Namen der Variablen aus der Steuerungsdatei. Es existiert teilweise eine Kurz- und eine Langform:

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
```

## 5. Erzeugen der fli4l Archive/Bootmedien

<code>--filesonly</code>	create only fli4l-files - do not create a boot-media
<code>--no-squeeze</code>	don't compress shell scripts
<code>--rebuild</code>	rebuild mkfli4l and related tools; needs make, gcc

Eine HD-Vorinstallation einer passend formatierten (FAT16/FAT32) CompactFlash im USB-Cardreader oder eines USB-Sticks ist über die Option `--hdinstallpath <dir>` möglich. Dieses können Sie *auf eigenes Risiko* zur Installation auf eine CompactFlash oder einen USB-Stick benutzen. Hierbei werden auf die angegebene Partition die nötigen Dateien des fli4l kopiert. Sie rufen dazu zunächst im fli4l-Verzeichnis

```
sh mkfli4l.sh --hdinstallpath <dir>
```

auf. Dabei werden die fli4l Dateien auf eine CF-Card oder USB-Stick kopiert.

Um die nächsten Schritte ausführen zu können, sind folgende Voraussetzungen zu erfüllen:

- `chmod 777 /dev/brain`
- superuser-Rechte
- installiertes `syslinux`
- installiertes `fdisk`

Durch das Script erfolgt eine Kontrolle, ob dieser Datenträger tatsächlich ein USB-Laufwerk ist und die erste Partition eine FAT-Partition ist. Anschliessend werden der Bootloader und die nötigen Dateien auf den angegebenen Datenträger kopiert. Sie erhalten eine Meldung über den Erfolg oder Misserfolg.

Nach dem Build müssen Sie

```
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
```

ausführen. Dann sollte die CF bzw. der USB-Stick bootfähig sein. Vergessen Sie nicht, den Datenträger auszuhängen (via `umount`).

Als letzter Optionsparameter kann ein alternatives Konfigurationsverzeichnis übergeben werden. Das normale Konfigurationsverzeichnis heißt `config` und liegt direkt im fli4l Wurzelverzeichnis. An diesem Ort legen alle fli4l Pakete die Konfigurationsdateien ab. Möchte man mehr

als eine Konfiguration verwalten, so erstellt man sich ein weiteres Verzeichnis, z.B. `hd.conf`, legt dort eine Kopie der Konfigurationsdateien ab und verändert diese den Anforderungen entsprechend. Hier einige Beispiele:

```
sh mkfli4l.sh --filesonly hd.conf
sh mkfli4l.sh --no-squeeze config.test
```

### 5.2. Erzeugen der fli4l Archive/Bootmedien unter Windows

Es wird das Tool ‘AutoIt3’ verwendet (<http://www.autoitscript.com/site/autoit/>). Dieses ermöglicht eine ‘grafische’ Ausgabe, sowie Dialoge, mit denen die in den folgenden Abschnitten beschriebenen Variablen beeinflusst werden können.

`mkfli4l.bat`

Das Build-Programm erkennt selbständig die unterschiedlichen [Bootvarianten](#) (Seite 24).

Der Aufruf von ‘mkfli4l.bat’ kann direkt aus dem Windows Explorer erfolgen, wenn man keine optionalen Parameter verwenden möchte.

Die Aktionen des Build-Programms werden durch verschiedene Mechanismen gesteuert:

- Konfigurationsvariable `BOOT_TYPE` aus der `<config>/base.txt`
- Konfigurationsdatei `<config>/mkfli4l.txt`
- Parameter des Build-Programmes
- Interaktive Einstellung in der GUI

An Hand der Konfigurationsvariable `BOOT_TYPE` (Seite 24) entscheidet sich, welche Aktion das Build-Programm ausführt:

- Erstellen eines bootfähigen fli4l CD-ISO-Images
- Bereitstellen der fli4l Dateien, zwecks Remote-Update
- Erzeugen der fli4l Dateien und direktes Remote-Update per SCP
- HD-pre-install einer passend formatierten CF im Cardreader
- usw.

Die Beschreibung der Variablen der Konfigurationsdatei `<config>/mkfli4l.txt` finden Sie im Kapitel [Steuerungsdatei mkfli4l.txt](#) (Seite 114).

#### 5.2.1. Kommandozeilenoptionen

Ein weiterer Steuerungsmechanismus ist das Anhängen von Optionsparametern an den Aufruf des Build-Programms auf der Kommandozeile. Die Steuerungsmöglichkeiten entsprechen denen der Steuerungsdatei `mkfli4l.txt`. Die Angabe von Optionsparametern überschreiben die Werte aus der Steuerungsdatei. Aus Komfortgründen unterscheiden sich die Namen der Optionsparameter von den Namen der Variablen aus der Steuerungsdatei. Es existiert teilweise eine Kurz- und eine Langform:

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]
```

Als letzter Optionsparameter kann ein alternatives Konfigurationsverzeichnis übergeben werden. Das normale Konfigurationsverzeichnis heißt `config` und liegt direkt im fli4l Wurzelverzeichnis. An diesem Ort legen alle fli4l Pakete die Konfigurationsdateien ab. Möchte man mehr als eine Konfiguration verwalten, so erstellt man sich ein weiteres Verzeichnis, z.B. `hd.conf`, legt dort eine Kopie der Konfigurationsdateien ab und verändert diese den Anforderungen entsprechend. Hier einige Beispiele:

```
mkfli4l.bat hd.conf
mkfli4l.bat -v
mkfli4l.bat --no-gui config.hd
```

### 5.2.2. Konfigurationsdialog – Einstellung des Konfigurationsverzeichnis

Im Hauptfenster wird die Einstellung des Konfigurationsverzeichnis angezeigt und es kann ein Fenster geöffnet werden zur Auswahl des Konfigurationsverzeichnis.

Zu beachten ist, dass eine Änderung des 'Config-Dir' alle Optionen auf die Werte setzt, die in der dortigen [Steuerungsdatei 'mkfli4l.txt'](#) (Seite 114) gesetzt bzw. als Kommandozeilenparameter übergeben wurden.

## 5. Erzeugen der fli4l Archive/Bootmedien

Findet mkfli4l.bat kein Verzeichnis fli4l-x.y.z\config oder in dem Verzeichnis keine Datei mit dem Namen 'base.txt' öffnet sich sofort das Fenster zur Auswahl des Konfigurationsverzeichnis. Dieses ermöglicht es auf einfache Weise im fli4l-Verzeichnis mehrere Konfigurationen zu verwalten.

Beispiel:

```
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-erstellen
```

### 5.2.3. Konfigurationsdialog – allgemeine Einstellungen



Abbildung 5.1.: Einstellungen

In diesem Dialog werden die Einstellungen für die Archiv/Bootmedienerstellung festgelegt:

## 5. Erzeugen der fli4l Archive/Bootmedien

- Build-Dir – Verzeichnis für die Archive/CD-Images/...
- BOOT\_TYPE – Anzeige des verwendeten/eingestellten BOOT\_TYPE – nicht änderbar
- Verbose – Aktivierung von zusätzlichen Ausgaben während der Erstellung
- Filesonly – es werden nur die Archive erstellt – kein bootmedium/kein Image
- Remoteupdate – Aktivierung des Remoteupdates per SCP

Mit der Schaltfläche **Aktuelle Einstellungen in mkfli4l.txt speichern** können die aktuell eingestellten Werte in der mkfli4l.txt gespeichert werden.

### 5.2.4. Konfigurationsdialog – Einstellungen für Remoteupdate



Abbildung 5.2.: Einstellungen für Remoteupdate

In diesem Dialog werden die Einstellungen für den Remoteupdate festgelegt:



## 5. Erzeugen der fli4l Archive/Bootmedien

- IP-Adresse oder Hostname
- Benutzername auf dem Remote-Host
- Remote-Pfad (default: /boot)
- Remote-Port (default: 22)
- zu verwendendes SSH-Keyfile (ppk-Format von Putty)

### 5.2.5. Konfigurationsdialog – Einstellungen für HD-pre-install



Abbildung 5.3.: Einstellungen für HD-pre-install

In diesem Dialog können die Optionen für den HD-pre-install auf einer entsprechend partitionierten und formatierten CompactFlash-Karte in einem USB-Reader eingestellt werden.  
Mögliche Optionen:

- HD-pre-install aktivieren

- Laufwerksbuchstabe der CF-Karte

Hinweis zur Partitionierung und Formatierung der CF: Für eine HD-Installation nach TYP A (siehe dazu Paket HD) muss auf der CF eine primäre aktive und formatierte FAT-Partition vorhanden sein. Möchte man weiterhin auch eine Datenpartition benutzen, wird zusätzlich eine Linux-Partition, die mit dem Dateisystem ext3 formatiert ist, sowie die Datei `hd.cfg` auf der FAT-Partition benötigt (hierzu sollten unbedingt die Hinweise im Paket HD beachtet werden).

### 5.3. Steuerungsdatei `mkfli4l.txt`

Seit fli4l-Version 2.1.9 existiert die Steuerungsdatei `<config>/mkfli4l.txt`. Durch sie werden z.B. vom Standard abweichende Verzeichnisse übergeben. Die Steuerungsdatei hat einen ähnlichen Aufbau wie die normalen fli4l Konfigurationsdateien. Alle Konfigurationsvariablen sind hier optional, d.h. sie müssen nicht in der Konfigurationsdatei vorkommen oder können als Kommentar gekennzeichnet werden.

**BUILDDIR** Standardwert: `'build'`

Legt fest, in welchem Verzeichnis die fli4l Dateien erzeugt werden sollen. Ist die Variable undefiniert, setzt `mkfli4l` unter Windows `'build'` relativ zum fli4l Wurzelverzeichnis ein und meint damit also das Verzeichnis `build` im fli4l Wurzelverzeichnis:

`Pfad/fli4l-x.y.z/build`

Unter \*nix setzt `mkfli4l` `<config>/build` ein und legt damit die generierten Dateien zusammen mit der Konfiguration ab.

Die konfigurierten Pfade in **BUILDDIR** müssen der jeweiligen Logik von Windows oder \*unix entsprechen. Werden relative Pfade gesetzt, wird der Pfad durch den Buildprozess passend zu Windows oder \*unix konvertiert.

**VERBOSE** Standardwert: `VERBOSE='no'`

Mögliche Werte sind `'yes'` oder `'no'`. Steuert die *Geschwätzigkeit* des Build Prozesses.

**FILESONLY** Standardwert: `FILESONLY='no'`

Mögliche Werte `'yes'` oder `'no'`. Hiermit kann das Erstellen eines Boot-Mediums abgeschaltet werden, es werden also nur die Dateien erzeugt –

**REMOTEUPDATE** Standardwert: `REMOTEUPDATE='no'`

Mögliche Werte `'yes'` oder `'no'`. Aktiviert das automatische Übertragen der erstellten Dateien mittels SCP auf den Router. Dieses setzt ein installiertes Paket SSHD (Seite ??) mit aktiviertem `scp` voraus. Siehe dazu auch die folgenden Variablen.

**REMOTEHOSTNAME** Standardwert: `REMOTEHOSTNAME=""`

Gibt den Ziel-Hostnamen für den SCP Datentransfer an. Sollte kein Name angegeben sein, wird dieser der Variable `HOSTNAME` (Seite 23) entnommen.

**REMOTEUSERNAME** Standardwert: `REMOTEUSERNAME='fli4l'`

Username für den SCP Datentransfer.

**REMOTEPATHNAME** Standardwert: REMOTEPATHNAME= '/boot'

Ziel-Pfad für den SCP Datentransfer.

**REMOTEPORT** Standardwert: REMOTEPORT= '22'

Zielpport für den SCP Datentransfer.

**SSHKEYFILE** Standardwert: SSHKEYFILE=""

Hier kann man eine SSH-Keydatei für den SCP-Remoteupdate angeben. Es kann somit ein Update ohne Angabe eines Passwortes erfolgen.

**REMOTEREMOUNT** Standardwert: REMOTEREMOUNT='no'

Mögliche Werte 'yes' oder 'no'. Wird hier 'yes' gesetzt, wird ein eventuell Readonly eingehängtes Bootdevice /boot für das Remoteupdate Readwrite gemountet um das Remoteupdate möglich zu machen.

**TFTPBOOTPATH** Pfad an dem das Netboot-Image abgelegt wird.

**TFTPBOOTIMAGE** Name des Netboot-Images.

**PXESUBDIR** Unterverzeichnis für die PXE-Dateien relativ zu TFTPBOOTPATH.

**SQUEEZE\_SCRIPTS** Aktiviert bzw. deaktiviert das Squeezten (Kommprimieren) von Skripten. Das Komprimieren eines Skripts mit Squeeze entfernt alle Kommentare und Zeileneinrückungen. Im Normalfall sollte hier immer der Standardwert 'yes' benutzt werden.

**MKFLI4L\_DEBUG\_OPTION** Es können zum Debuggen zusätzliche Optionen an das mkfli4l-Programm (Seite ??) übergeben werden.

## 6. Anbindung von PCs im LAN

Für jeden Rechner im LAN ist einzustellen:

1. IP-Adresse (siehe [IP-Adresse](#))
2. Name des Rechners plus Wunsch-Domain-Name (siehe [Rechnername und Domain](#))
3. Standard-Gateway (siehe [Gateway](#))
4. IP-Adresse des DNS-Servers (siehe [DNS-Server](#))

### 6.1. IP-Adresse

Die IP-Adresse muss im gleichen Netz wie die IP-Adresse des fli4l-Routers (auf Ethernet-Seite) liegen, also z.B. 192.168.6.2, wenn der fli4l die Adresse 192.168.6.1 hat. Kein Rechner darf die gleiche IP-Adresse haben, weshalb man am besten (nur) die letzte Zahl ändert. Auch ist darauf zu achten, dass man hier die gleiche IP-Adresse angibt, wie man es für diesen Rechner in der Datei config/base.txt angegeben hat.

### 6.2. Rechnername und Domain

Der Name des Rechners ist dann z.B. "mein-pc", die Domain "lan.fli4l".

**Wichtig:** Die im PC eingestellte Domain muss identisch mit der gewählten Domain im fli4l-Rechner sein, wenn man den fli4l-Router als DNS-Server verwenden will. Sonst kann es im Netz erhebliche Probleme geben.

Grund: Windows-Rechner suchen regelmäßig nach Rechnern mit dem Namen ihrer Arbeitsgruppe: WORKGROUP.meine-domain.fli4l. Ist dies nicht die in fli4l eingestellte Domain (hier: meine-domain.fli4l), wird fli4l versuchen, diese Anfrage durch Weiterleiten ins Internet zu beantworten ...

Einzutragen ist die Domain in den TCP/IP Einstellungen des Rechners.

#### 6.2.1. Windows 2000

Für Windows 2000 findet man das unter:

Start ⇒

Einstellungen ⇒

Systemsteuerung ⇒

Netzwerk- und DFÜ-Verbindungen ⇒

LAN-Verbindung ⇒

Eigenschaften ⇒

Internetprotokoll (TCP/IP) ⇒

## 6. Anbindung von PCs im LAN

Eigenschaften ⇒  
Erweitert... ⇒  
DNS ⇒  
DNS-Suffix hinzufügen ⇒

“lan.fli4l” (bzw. die eingestellte domain) eingeben (ohne “”) ⇒ OK drücken.

### 6.2.2. NT 4.0

Start ⇒  
Einstellungen ⇒  
Systemsteuerung ⇒  
Netzwerk ⇒  
Protokolle ⇒  
TCP/IP ⇒  
Eigenschaften ⇒  
DNS ⇒

- Hostname eintragen (eigener Rechnername)
- Domäne eintragen (wie in config/base.txt)
- IP-Adresse vom fli4l-Router hinzufügen
- DNS-Suffix hinzufügen (Domäne hinzufügen – siehe 2 Zeilen höher)

### 6.2.3. Win95/98

Start ⇒  
Einstellungen ⇒  
Systemsteuerung ⇒  
Netzwerk ⇒  
Konfiguration ⇒  
TCP/IP (jenes, das an die Netzwerkkarte zum Router  
angebunden ist) ⇒  
Eigenschaften ⇒  
DNS-Konfiguration:  
DNS aktivieren und bei “Domäne:” dann “lan.fli4l” eingeben (ohne “”).

### 6.2.4. Windows XP

Für Windows XP findet man das unter:

Start ⇒  
Einstellungen ⇒  
Systemsteuerung ⇒  
Netzwerkverbindungen ⇒  
LAN-Verbindung ⇒  
Eigenschaften ⇒

Internetprotokoll (TCP/IP) ⇒  
Eigenschaften ⇒  
Erweitert... ⇒  
DNS ⇒  
DNS-Suffix für diese Verbindung ⇒

“lan.fli4l” (bzw. die eingestellte domain) eingeben (ohne “”) ⇒OK drücken.

### 6.2.5. Windows 7

Für Windows 7 findet man das unter:

Windows Button (ex. Start) ⇒  
Systemsteuerung ⇒  
Netzwerk und Internet ⇒  
Netzwerk- und Freigabecenter ⇒  
LAN-Verbindung ⇒  
Eigenschaften ⇒  
Internetprotokoll Version 4 (TCP/IPv4) ⇒  
Eigenschaften ⇒  
Erweitert... ⇒  
DNS ⇒  
DNS-Suffix für diese Verbindung ⇒

“lan.fli4l” (bzw. die eingestellte domain) eingeben (ohne “”) ⇒OK drücken.

### 6.2.6. Windows 8

Für Windows 8 findet man das unter:

Gleichzeitig Windows- und X-Taste drücken ⇒  
Systemsteuerung ⇒  
Netzwerk und Internet ⇒  
Netzwerk- und Freigabecenter ⇒  
Ihr Netzwerk wählen (Ethernet oder WLAN) ⇒  
Eigenschaften ⇒  
Internetprotokoll Version 4 (TCP/IPv4) ⇒  
Eigenschaften ⇒  
Erweitert... ⇒  
DNS ⇒  
DNS-Suffix für diese Verbindung ⇒

“lan.fli4l” (bzw. die eingestellte domain) eingeben (ohne “”) ⇒OK drücken.

## 6.3. Gateway

Die Angabe des Standard-Gateways ist unbedingt erforderlich, denn ohne die Angabe der richtigen IP-Adresse an dieser Stelle funktioniert nichts. Es muß hier die IP-Adresse des fli4l-

Routers (auf Ethernet-Seite) angegeben werden, also z.B. 192.168.6.4 entsprechend der IP-Adresse, die hier in der Datei config/base.txt für den fli4l-Router angegeben wurde.

Es ist falsch, den fli4l-Router als Proxy in der Windows- oder Browser- Konfiguration einzutragen – außer man setzt ein Proxy auf dem fli4l-Router ein. Im Normalfall ist fli4l kein Proxy, daher bitte *nicht* fli4l als Proxy angeben!

### 6.4. DNS-Server

Als IP-Adresse des DNS-Servers gibt man nicht die Adresse des Provider-DNS-Servers an, sondern die des fli4l-Routers (Ethernet), da dieser nun selbst Anfragen beantworten kann bzw. diese bei Unkenntnis ins Internet weiterleitet.

Mit der Konstruktion von fli4l als DNS-Server werden viele von den Windows-PCs ausgeführten Anfragen nicht ins Internet weitergeroutet, sondern werden direkt vom fli4l-Router beantwortet.

### 6.5. Verschiedenes

Die Punkte 1 bis 4 brauchen bei konfiguriertem DHCP-Server nicht eingetragen zu werden, da dann der fli4l-Router die notwendigen Daten automatisch übermittelt.

**Internetoptionen:** Bei Verbindungen muß “keine Verbindung wählen” ausgewählt sein. Bei Einstellungen für lokales Netzwerk(LAN): es darf hier NICHTS angegeben werden (es sei dann es wird OPT\_Proxy verwendet). Beides sind Standardeinstellungen, die im Normalfall nicht geändert werden müssen.

## 7. Client-/Server-Schnittstelle imon

### 7.1. imon-Server imond

imond ist ein netzwerkfähiges Server-Programm, welches bestimmte Anfragen beantwortet oder auch Kommandos zur Steuerung des Routers entgegennehmen kann.

Ausserdem steuert imond das Least-Cost-Routing. Dazu verwendet er eine Konfigurationsdatei `/etc/imond.conf`, welche beim Booten automatisch aus den `ISDN_CIRC_x_XXX`-Variablen der Datei `config/isdn.txt` und anderen über ein Shell-Script erzeugt wird.

imond läuft permanent als Daemon und horcht gleichzeitig auf TCP/IP-Port 5000 und Device `/dev/isdninfo`.

Folgende Kommandos sind über den TCP/IP-Port 5000 möglich:

Der TCP/IP-Port 5000 ist nur vom maskierten LAN aus erreichbar. Standardmäßig wird ein Zugriff von aussen über die Firewall-Konfiguration abgeblockt.

Imond unterstützt zwei Benutzerebenen: den User- und den Admin-Modus. Für beide Ebenen kann ein Passwort gesetzt werden mittels `IMOND_PASS` bzw. `IMOND_ADMIN_PASS`. Dadurch werden die imon-Clients von imond gezwungen, eine Password-Abfrage durchzuführen und anschließend das Password an imond zu übertragen. Solange dieses Password nicht übermittelt wurde, nimmt imond nur die beiden Kommandos "pass" und "quit" entgegen. Alle anderen werden mit einem Fehler zurückgewiesen.

Möchte man das weiter einschränken, z.B. den Zugriff nur von nur einem PC erlauben, muss die Firewall-Konfiguration angepasst werden.

Die Befehle

```
enable/disable/dialmode  dial/hangup  route  reboot/halt
```

können durch die Konfigurationsvariablen `IMOND_XXX` global ein- oder abgeschaltet werden (s. Kapitel "Konfiguration").

Mit einem Unix/Linux-Rechner (oder einem Windows-Rechner in der DOS-Box) kann man das Ganze leicht ausprobieren:

Nach Eingabe von

```
telnet fli4l 5000          \# oder entsprechender Name des fli4l-Routers
```

kann man direkt die oben aufgeführten Kommandos eingeben und sich die Ausgabe anschauen.

Zum Beispiel bekommt man mit "help" die Hilfe angezeigt, mit "quit" wird die Verbindung zum imond abgebaut.

#### 7.1.1. Least-Cost-Routing – Funktionsweise

imond konstruiert aus der Konfigurationsdatei `/etc/imond.conf` (welche wiederum beim Booten aus den Konfigurationsvariablen `ISDN_CIRC_x_TIMES` usw. erstellt wird), eine zeitabhängige



### Admin-Befehle

addlink ci-index	Channel zum Circuit hinzufügen (Channel-Bundling)
adjust-time seconds	Ändert die Uhrzeit des Routers um die angegebenen Sekunden
delete filename pw	Löscht die Datei auf dem Router
hup-timeout #ci-index [value]	Anzeigen bzw. Setzen des HUP-Timeout für ISDN-Circuits
removelink ci-index	Zusätzlichen Channel wieder entfernen
reset-telmond-log-file	Löschen der telmond-Protokolldatei
reset-imond-log-file	Löschen der imond-Protokolldatei
receive filename #bytes pw	Eine Datei auf den Router übertragen. Dazu quittiert imond den Befehl mit einem ACK (0x06). Danach wird die Datei in 1024er-Blöcken übertragen, die imond auch jeweils mit einem ACK bestätigt. Als letztes übermittelt imond noch ein OK.
send filename pw	Wenn das Passwort stimmt und die Datei existiert, liefert imond ein OK #bytes. Anschliessend überträgt imond die Datei in 1024er Blöcken, die jeweils mit einem ACK (0x06) bestätigt werden müssen. Als letztes liefert imond noch ein OK.
support pw	Liefert den Status/Konfiguration vom Router
sync	Synchronisiert den Cache von gemounteten Laufwerken

### Admin- oder User-Befehle

dial	Wählt den Provider an (Default-Route-Circuit)
dialmode [auto manual off]	Liefert bzw. setzt den Dialmode
disable	Hängt ein und setzt dialmode auf "off"
enable	Setzt dialmode auf "auto"
halt	Führt den Router sauber herunter
hangup [#channel-id]	Hängt ein
poweroff	Führt den Router herunter und schaltet ab
reboot	Reboot vom i4l-Router!
route [ci-index]	Setzen Default-Route auf Circuit X (0=automatisch)

## User-Befehle

channels	Ausgabe Anzahl der verfügbaren ISDN-Kanäle
charge #channel-id	Ausgabe der Online-Kosten für einen Channel
chargetime #channel-id	Online-Zeit unter Berücksichtigung des Taktes
circuit [ci-index]	Ausgabe eines Circuit-Namens
circuits	Ausgabe Anzahl der Default-Route-Circuits
cpu	Liefert die Auslastung der CPU in Prozent
date	Ausgabe Datum/Uhrzeit
device ci-index	Liefert das Device des Circuits
driverid #channel-id	Ausgabe Driver-Id für Channel X
help	Ausgabe Hilfe
inout #channel-id	Ausgabe der Richtung (incoming/outgoing)
imond-log-file	Ausgabe imond-Protokolldatei
ip #channel-id	Ausgabe der IP
is-allowed command	Ausgabe, ob Befehl konfiguriert/gültig ist Mögliche Befehle: dial dialmode route reboot  imond- log telmond-log mgetty-log
is-enabled	Ausgabe, ob dialmode auf off (0) oder auto (1)
links ci-index	Ausgabe Anzahl momentaner Channel 0, 1 oder 2, 0 heisst: Kein Channel-Bundling möglich
log-dir imond telmond mgetty	Liefert das Logverzeichnis
mgetty-log-file	Ausgabe mgetty-Protokolldatei
online-time #channel-id	Ausgabe Online-Zeit der akt. Verbindung in hh:mm:ss
pass [password]	Abfrage, ob Password nötig bzw. Password- Eingabe 1 Userpassword gesetzt 2 Adminpassword gesetzt 4 imond befindet sich im Admin-Modus
phone #channel-id	Ausgabe Telefonnummer/Name des "Gegners"
pppoe	Liefert die Anzahl der pppoe-Devices (also 0 oder 1)
quantity #channel-id	Liefert die übertragenen Datenmengen (in Byte)
quit	Beenden der Verbindung zu imond
rate #channel-id	Ausgabe Übertragungsraten (incoming/outgoing in B/sec)
status #channel-id	Ausgabe Status für Channel X
telmond-log-file	Ausgabe telmond-Protokolldatei
time #channel-id	Ausgabe Summe Online-Zeiten, Format hh:mm:ss
timetable [ci-index]	Ausgabe der Zeittabelle für LC-Routing
uptime	Ausgabe der Uptime des Routers in Sekunden
usage #channel-id	Ausgabe Art der Verbindung, mögliche Antworten: Fax, Voice, Net, Modem, Raw
version	Ausgabe der Protokoll- und Programm-Version

## 7. Client-/Server-Schnittstelle imon

Tabelle (Time-Table). Diese umfasst eine komplette Kalenderwoche im 1-Stunden-Raster (168 Stunden = 168 Bytes). Die Tabelle setzt sich jedoch lediglich aus Circuits zusammen, für die eine Default-Route definiert ist.

Mit dem imond-Kommando "timetable" kann man sich diese Tabelle anschauen.

Hier ein Beispiel:

Nehmen wir an, dass 3 Circuits definiert wurden, nämlich:

```
CIRCUIT_1_NAME='Addcom'
CIRCUIT_2_NAME='AOL'
CIRCUIT_3_NAME='Firma'
```

wobei lediglich die ersten beiden Circuits mit Default-Routen belegt sind, also die entsprechenden Variablen ISDN\_CIRC\_x\_ROUTE den Wert '0.0.0.0' haben.

Wenn die dazugehörigen Variablen ISDN\_CIRC\_x\_TIMES folgendermaßen aussehen:

```
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'
```

dann wird daraus folgende Datei /etc/imond.conf:

#day	hour	device	defroute	phone	name	charge	ch-int
Mo-Fr	09-18	ipp0	no	010280192306	Addcom	0.0388	60
Mo-Fr	18-09	ipp0	yes	010280192306	Addcom	0.0248	60
Sa-Su	00-24	ipp0	yes	010280192306	Addcom	0.0248	60
Mo-Fr	09-18	ipp1	yes	019160	AOL 0.019	180	
Mo-Fr	18-09	ipp1	no	019160	AOL 0.049	180	
Sa-Su	09-18	ipp1	no	019160	AOL 0.019	180	
Sa-Su	18-09	ipp1	no	019160	AOL 0.049	180	
Mo-Fr	09-18	isd2	no	0221xxxxxxx	Firma	0.08	90
Mo-Fr	18-09	isd2	no	0221xxxxxxx	Firma	0.03	90
Sa-Su	00-24	isd2	no	0221xxxxxxx	Firma	0.03	90

imond erstellt dann im Speicher folgende Time-Table – hier die Ausgabe über das imond-Kommando "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	0.0.0.0	ipp0	0.0388	60
2	AOL	0.0.0.0	ipp1	0.019	180
3	Firma	0.0.0.0	isd2	0.08	90

## 7. Client-/Server-Schnittstelle imon

1	Addcom		no	ipp0	0.0388	60
2	Addcom		yes	ipp0	0.0248	60
3	Addcom		yes	ipp0	0.0248	60
4	AOL	yes	ipp1	0.0190	180	
5	AOL	no	ipp1	0.0490	180	
6	AOL	no	ipp1	0.0190	180	
7	AOL	no	ipp1	0.0490	180	
8	Firma		no	isd2	0.0800	90
9	Firma		no	isd2	0.0300	90
10	Firma		no	isd2	0.0300	90

Für den Circuit 1 (Addcom) sind also drei Zeitbereiche (1-3) eingetragen, für Circuit 2 (AOL) vier Zeitbereiche (4-7) und für den letzten drei Zeitbereiche (8-10).

In der Time-Table werden jeweils die Indices ausgegeben, welche für die jeweilige Stunde gültig sind. Hier tauchen lediglich die Indices 2-4 auf, da alle anderen keine LC-Default-Routen sind.

Sieht man in der Tabelle irgendwo Nullen, gibt es Lücken in den ISDN\_CIRC\_X\_TIMES-Werten. Dann existiert zu diesen Zeiten keine Default-Route, Internet-Zugang abgeknipst!

Beim Programmstart ermittelt imond zunächst den Wochentag und die aktuelle Stunde. Anschließend wird dann über die Time-Table der Index ermittelt und damit dann auch der entsprechende Circuit. Auf diesen wird dann die Default-Route gesetzt.

Bei Zustandsänderungen der Channels, z.B. Wechsel von online nach offline – jedoch spätestens nach 1 Minute – geht das Spiel von neuem los: Zeit ermitteln, Lookup in Tabelle, Default-Route-Circuit ermitteln.

Ändert sich der aktuell verwendete Circuit, z.B. montags um 18:00 Uhr, wird die alte Default-Route gelöscht, eine vielleicht bestehende Verbindung beendet (sorry...) und anschließend die Default-Route auf den neuen Circuit gesetzt. Dies kann von imond bis zu 60 Sekunden später bemerkt werden, also wird spätestens um 18:00:59 umgeschaltet.

Bei Circuits, die keine Default-Route belegen, ändert sich überhaupt nichts. Hier wird der Inhalt von ISDN\_CIRC\_x\_TIMES lediglich zur Berechnung der Telefonkosten verwendet. Diese können dann relevant sein, wenn man über den Client imonc das LC-Routing temporär ausschaltet und einen Circuit manuell wählt.

Man kann sich jedoch auch die Tabellen für andere Zeitbereich-Indices (im Beispiel von 1 bis 10) anschauen, auch die der "Non-LC-Default-Route-Circuits".

Kommando:

```
timetable index
```

Beispiel:

```
telnet fli41 5000
timetable 5
quit
```

Die Ausgabe sieht dann so aus:

	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	5	5	5	5	5	5
We	5	5	5	5	5	5	5	5	5	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	5	5	5	5	5	5
Fr	5	5	5	5	5	5	5	5	5	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

No.	Name	DefRoute	Device	Ch/Min	ChInt
5	AOL	no	ipp1	0.0490	180

Alles klar?

Mit dem imond-Kommando "route" kann das LC-Routing ein- und ausgeschaltet werden. Bei Angabe eines positiven Circuit-Indices (1...N) wird die Default-Route auf den angegebenen Circuit gelegt. Ist der Index 0, wird das LC-Routing wieder aktiviert und der Circuit automatisch ausgewählt.

### 7.1.2. Zur Berechnung der Onlinekosten

Das ganze Modell zur Berechnung der Onlinekosten funktioniert nur korrekt, wenn der Zeittakt für einen Circuit (Variable `ISDN_CIRC_x_CHARGEINT`) über die ganze Woche konstant ist. Dies ist im Normalfall bei Internet-Providern die Regel. Wählt man sich jedoch über die Telekom (ich meine nicht T-Online!) z.B. in sein Firmennetz ein, gilt das als ganz normales Telefongespräch. Und da wechselt ab 18:00 der Takt von 90 Sekunden auf 4 Minuten (Stand Juni 00). Deshalb ist die Definition von

```
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'
```

eigentlich nicht ganz korrekt. Es sind zwar abends umgerechnet auf die Minute 3 Pfennig (4 Minuten kosten 12 Telekom-Pfennige), jedoch ist der Takt falsch. Deshalb können bei der Kostenanzeige Differenzen zu den tatsächlichen Zahlen auftreten.

Hier ist ein Tipp, wie verschieden lange Taktzeiten dennoch richtig berücksichtigt werden können (auch wichtig für `ISDN_CIRC_x_CHARGEINT`): Man definiert einfach 2 Circuits, einen für tagsüber mit `ISDN_CIRC_1_CHARGEINT='90'` und den anderen mit `ISDN_CIRC_2_CHARGEINT='240'`. Natürlich muss man dann auch noch `ISDN_CIRC_x_TIMES` entsprechend wählen, damit tagsüber Circuit 1 und abends Circuit 2 verwendet wird.

Wie gesagt: Bei Nutzung von Verbindungen zu Internet-Providern gibt es das Problem nicht, weil dort der Zeittakt immer konstant ist und lediglich die Kosten pro Minute wechseln (oder gibt es sowas doch??? Ich traue T-\* alles zu :-).

## 7.2. Windows-Client imonc.exe

### 7.2.1. Einleitung

Das Gespann imond auf dem Router und imonc auf dem Client beherrschen zwei Benutzermodi: den User- und den Adminmodus. Im Adminmodus sind alle Steuerelemente aktiviert. Im Usermodus steuern die Variablen `IMOND_ENABLE` (Seite 81), `IMOND_DIAL` (Seite 80), `IMOND_ROUTE` (Seite 80) und `IMOND_REBOOT` (Seite 80) ob die jeweiligen Funktionen im Usermodus zur Verfügung stehen. Sind alle diese Variablen auf 'no' gesetzt, bedeutet dies für die Überblick-Seite,

dass alle Buttons bis auf den Exit- und den Admin-Mode-Button deaktiviert sind. Die Entscheidung, ob der User- oder Admin-Modus benutzt wird, wird anhand des übermittelten Passwortes getroffen. Über den Button Admin-Mode, der sich in der Statusleiste befindet, kann jederzeit unter Eingabe des Admin-Passwortes vom User- zum Admin-Modus gewechselt werden. Um wieder zurück zu wechseln, muss imonc beendet und neu gestartet werden.

Sobald imonc gestartet ist, wird ein zusätzliches Tray-Icon angezeigt, welches den Verbindungsstatus der vorhandenen Kanäle anzeigt.

Die Farben bedeuten:

**Rot** : Offline

**Gelb** : Es wird gerade eine Verbindung aufgebaut

**Hellgrün** : Online und Traffic auf dem Kanal

**Dunkelgrün** : Online und so gut wie kein Traffic auf dem Kanal

Ein etwas vom Windows-Standard abweichendes Verhalten zeigt imonc, wenn der Minimieren-Button in der Titelleiste angeklickt wird. Daraufhin minimiert sich imonc in den Systemtray und es bleibt nur noch das Tray-Icon neben der Uhr übrig. Ein Doppelklick mit der linken Maustaste auf das Tray-Symbol holt das imonc-Fenster wieder in den Vordergrund. Mit der rechten Maustaste besteht auch die Möglichkeit über das Kontextmenü, die wichtigsten imonc-Kommandos direkt auszuwählen, ohne imonc wieder auf den Bildschirm zu holen.

Viele Eigenschaften (darunter auch alle Spaltenbreiten der StringGrids) speichert imonc in der Registry, damit imonc so an die eigenen Bedürfnisse angepasst werden kann. Imonc speichert die Informationen in dem Registry-Schlüssel HKCU\Software\fli4l.

Bestehen trotz sorgfältigen Lesens der Dokumentation noch Probleme in Bezug auf imonc oder auch des Routers selber, die man z.B. in der Newsgroup posten möchte, ist es sinnvoll, auf der Über-Seite des imonc den Punkt SystemInfo auszuwählen und dort den Punkt Support Infos. Daraufhin wird das Router-Passwort abgefragt (nicht das imonc-Passwort!). Imonc erstellt dann eine Datei fli4lsup.txt, welche alle wichtigen Informationen bezüglich des Routers und imonc beinhaltet. Diese Datei kann auf explizite Nachfrage in die Newsgroup gepostet werden, so dass deutlich bessere Chancen auf rasche Hilfe bestehen.

Nähere Details betreffend der Entwicklung des Windows-Clients imonc findet man auf der Homepage vom Windows ImonC-Seiten <http://www.imonc.de/>. Hier kann man sehen, welche neuen Features und Bug-Fixes in der nächsten Version von imonc enthalten sein werden. Ausserdem gibt es dort den neusten imonc, wenn dieser nicht schon in der fli4l-Distribution enthalten ist.

### 7.2.2. Startparameter

ImonC benötigt den Namen oder die IP-Adresse des fli4l-Routers. Standardmäßig versucht das Programm, eine Verbindung mit dem Rechner "fli4l" herzustellen. Wenn dieser im DNS korrekt eingetragen ist, sollte es also direkt funktionieren. Ansonsten kann man in der Verknüpfung folgende Parameter übergeben:

- /Server:IP oder Hostname des Routers (Kurzform: /S:IP oder Hostname)
- /Password:Passwort (Kurzform: /P:Password)

- `/log` Die Logging-Option zum Protokollieren der Kommunikation zwischen imonc und imond. Ist diese Option eingeschaltet, wird beim Beenden von imonc eine Datei imonc.log geschrieben. Diese Datei beinhaltet die gesamte Kommunikation zwischen Router und Client und wird darum sehr groß. Deshalb sollte dieser Startparameter nur gesetzt werden, wenn Probleme bestehen.
- `/iport:Portnummer` Die Portnummer auf die imond lauscht. Default: 5000
- `/tport:Portnummer` Port auf dem telmond lauscht. Default: 5001
- `/rc:"Command"` Das hier angegebene Kommando wird ohne weitere Überprüfung an den Router übertragen und anschliessend imonc beendet. Sollen mehrere Kommandos gleichzeitig ausgeführt werden, müssen diese durch Semikolons getrennt werden. Damit es funktioniert, muss ein gesetztes imond-Passwort mit übergeben werden, da keine Abfrage des Passwortes erfolgt. Die möglichen Kommandos sind beim imond dokumentiert, siehe Kapitel 8.1. Zusätzlich zu den dort aufgeführten Befehlen gibt es noch den Befehl `timesync`. Dieser bewirkt, dass die Uhrzeit des Clients mit der des Routers synchronisiert wird. Der Befehl `dialtimesync` wird nicht mehr unterstützt da er sich als „dial; timesync“ schreiben lässt.
- `/d:"fli4l-Directory"` Hiermit kann das fli4l-Directory per Startparameter übergeben werden. Interessant wenn man mit mehreren fli4l-Versionen herumspielt
- `/wait` Wenn der Hostname nicht aufgelöst werden kann, beendet sich imonc nicht mehr – erneuter Verbindungsaufbau durch Doppelclick auf das TrayIcon
- `/nostartcheck` Dieser schaltet die Überprüfung ab, ob imonc bereits läuft. Nur sinnvoll, wenn mehrere, unterschiedliche fli4l-Router in einem Netz überwacht werden sollen. Bei weiteren Instanzen werden die eingebauten Syslog- und E-Mail-Funktionalitäten deaktiviert.

Usage (einzutragen in der Verknüpfung):

```
X:\...imonc.exe [/Server:Host] [/Password:Passwort] [/iport:Portnummer]
                [/log] [/tport:Portnummer] [/rc:"Command"]
```

Beispiel mit IP-Adresse:

```
C:\wintools\imonc /Server:192.168.6.4
```

oder mit Namen und Passwort:

```
C:\wintools\imonc /S:fli4l /P:geheim
```

oder mit Namen, Passwort und Routerkommando:

```
C:\wintools\imonc /S:fli4l /P:geheim /rc:"dialmode manual"
```

### 7.2.3. Seite Überblick

Der Windows-Client fragt einige imond-Informationen über die bestehenden Verbindungen ab und bereitet sie im Anzeigefenster auf. Neben generellen Statusinformationen wie Uptime des Router oder auch der Uhrzeit sowohl lokal wie auch vom Router selber, werden für jede bestehende Verbindung die folgenden Informationen angezeigt:

Status	Verbindungsaufbau/Online/Offline
Name	Telefonnummer des Gegners oder Circuit-Name
Richtung	Zeigt an, ob es sich um eine eingehende oder ausgehende Verbindung handelt
IP	Die IP, die man zugewiesen bekommen hat
IBytes	Empfangene Bytes
OBytes	Gesendete Bytes
Online-Zeit	Aktuelle Online-Zeit
Zeit	Summe aller Online-Zeiten
KZeit	Summe Online-Zeiten unter Berücksichtigung des Zeittaktes
Kosten	Berechnete Kosten

Die Daten werden standardmäßig alle 2 Sekunden aktualisiert. Im Kontextmenü dieser Übersicht besteht die Möglichkeit für jeden vorhandenen Kanal, mit dem der Router gerade online ist, sowohl die zugewiesene IP in die Zwischenablage zu kopieren, als auch den Kanal gezielt auflegen zu können. Letzteres ist für den Fall interessant, dass mehrere unterschiedliche Verbindungen bestehen, z.B. eine um im Internet zu surfen und eine andere zur Firma, und gezielt eine dieser Verbindungen getrennt werden soll.

Ist zusätzlich auf dem fli4l-Router der telmond-Prozess aktiv, kann imonc zusätzlich Informationen über eingehende Telefonanrufe (nämlich anrufende und angerufene MSN) anzeigen. Der letzte eingegangene Telefonanruf wird oberhalb der Buttons angezeigt. Ein Protokoll der eingegangenen Telefonanrufe erhält man durch Anzeige der Seite Anrufe.

Mit den sechs Buttons im imonc können folgende Kommandos angewählt werden:

Button	Beschriftung	Funktion
1	Verbinden/Trennen	Wählen/Einhängen
2	Add link/Rem link	Kanäle bündeln: ja/nein – dieses Feature steht nur im Admin-Mode zur Verfügung
3	Reboot	fli4l neu booten!
4	PowerOff	fli4l sauber runterfahren und anschliessend ausschalten
5	Halt	fli4l sauber runterfahren, um ihn anschliessend sicher ausschalten zu können
6	Beenden	Client beenden

Die ersten fünf Kommandos können in der Konfigurationsdatei des fli4l-Routers config/base.txt für den User-Modus einzeln ein- und ausgeschaltet werden. Im Admin-Modus sind immer alle aktiviert. Die Auswahl Dialmode steuert das Wahlverhalten des Routers:



Auto	Der Router baut automatisch eine Verbindung auf dem entsprechenden Circuit auf, wenn eine Anfrage aus dem lokalen Netz eintrifft.
Manuell	Der Benutzer muss selber die Verbindung aufbauen.
Aus	Es ist weder manuell noch automatisch möglich, eine Verbindung aufzubauen. Der Dial-Button ist dann deaktiviert.

Bleibt noch anzumerken, dass fli4l standardmäßig selbständig rauswählt, wenn man mit seinem Rechner in's Internet will. Man muss also eigentlich nie den Verbinden-Button drücken ...

Es besteht auch die Möglichkeit, den Default-Route-Circuit manuell zu wechseln, also das automatische LCR-Routing ein- und auszuschalten. Dafür ist in der Windows-Version von imonc die Auswahlliste "Default Route" vorgesehen. Ausserdem kann man die Hangup-TimeOut-Zeit jetzt auch über imonc direkt konfigurieren. Dazu dient der Button Config neben der Default Route. Dort werden alle konfigurierten Circuits des Routers angezeigt. Der Wert in der Spalte Hup-timeout kann für ISDN-Circuits direkt im StringGrid editiert werden (funktioniert bis dato noch nicht für DSL).

Einen Überblick über das LCR-Routing findet man auf der Seite Admin/TimeTable. Dort sieht man, welchen Circuit imond zu welcher Zeit automatisch auswählt.

### 7.2.4. Config-Dialog

Der Konfigurationsbereich ist über den Button Config in der Statuszeile erreichbar. Das aufgehende Fenster ist dann in die folgenden Bereiche unterteilt:

- Der Bereich Allgemein:
  - Aktualisierungsintervall: Hier wird eingestellt, wie oft die Seite Überblick aktualisiert werden soll.
  - Zeit beim Programmstart synchronisieren: Übernimmt beim Starten des Client die Zeit und das Datum des Routers als lokale Zeit. Diese Funktion kann auch manuell mit dem Button Synchronisieren auf der Überblicks-Seite aufgerufen werden.
  - Minimiert starten: Startet das Programm direkt minimiert, man sieht nur das Icon neben der Uhr.
  - Zusammen mit Windows starten: Hier kann man angeben, ob der Client direkt beim Starten von Windows mit gestartet werden soll. In dem Feld Parameter kann man die nötigen Start-Parameter angeben.
  - News von fli4l.de abholen: Sollen die News, die auf der fli4l-Homepage in der News-Sektion angezeigt werden, auch vom imonc geholt und angezeigt werden? Die Schlagzeilen werden dann in der Statusbar angezeigt. Ausserdem wird dann eine neue Seite News angezeigt, in der die kompletten Meldungen angezeigt werden.
  - Logdatei für Verbindungen: Den Dateinamen, den man hier angeben kann, wird dazu benutzt, die Verbindungs-Liste unter diesem Namen lokal auf dem Rechner abzuspeichern.
  - TimeOut für Router zum antworten: Wie lange soll auf eine Antwort der Routers gewartet werden, bevor angenommen wird, dass die Verbindung nicht mehr besteht.
  - Sprache: Hier kann die Sprache des imoncs ausgewählt werden.

- Router Befehle bestätigen: Ist dieses Feature aktiviert, müssen alle Router-beeinflussenden Kommandos, wie zum Beispiel Reboot, Hangup . . . generell bestätigt werden.
  - Auflegen auch bei Traffic: Soll kein Hinweis erfolgen, wenn die Verbindung beendet wird und noch Traffic auf der Leitung ist.
  - Automatisch Verbindung zum Router aufbauen: Soll, wenn die Verbindung zum Router unterbrochen wird (z.B. durch einen Neustart des Routers), automatisch probiert werden, die Verbindung wieder herzustellen.
  - Fenster in System Tray minimieren: Soll imonc beim Klicken auf den Beenden-Button in der Titelleiste sich in den System-Tray neben der Uhr minimieren anstatt zu beenden.
- Der Unterbereich Proxy: Hier kann ein Proxy für die http-Anfragen des imoncs definiert werden. Dieser wird dann zur Zeit für das Holen der News benutzt.
    - Proxy-Unterstützung für Http-Anfragen aktivieren: Soll ein Proxy benutzt werden
      - \* Adresse: Die Adresse des Proxy-Servers
      - \* Port: Die Portnummer des Proxy-Server (default: 8080)
  - Der Unterbereich TrayIcon: Hier können die Farben des TrayIcons neben der Uhr an die eigene Bedürfnisse angepasst werden. Weiterhin kann ausgewählt werden, dass der aktuelle Dialmode als farblicher Hintergrund des TrayIcons dargestellt wird.
  - Der Bereich Anrufe: Die Position des Call Notification-Fensters wird in der Registry gespeichert, so dass man sich das Fenster an die Position schieben kann, wo man es haben möchte. Es erscheint anschliessend immer wieder an dieser Stelle.
    - Aktualisierung: Hier kann ausgewählt werden, wie imonc über neue Anrufe informiert wird. Es gibt drei verschiedene Möglichkeiten. Diese erste besteht darin, regelmäßig den telmond-Dienst auf dem Router abzufragen. Eine weitere Möglichkeit besteht in der Auswertung der Syslog-Meldungen. Diese Variante ist der ersten vorzuziehen – dazu muss natürlich der Syslog-Client des imonc aktiviert sein. Wird imonc mit einem routenden eisfair eingesetzt, bietet sich noch die Möglichkeit das Capi2Text-Paket zur Anrufsignalisierung zu benutzen.
    - Führende Null wegen Telefonanlage löschen: Telefonanlage setzen manchmal eine zusätzliche Null vor die Rufnummer des Anrufer. Diese kann mit dieser Option unterdrückt werden.
    - Eigene Vorwahl: Hier kann die eigene Vorwahl hinterlegt werden. Wann dann ein Anruf mit gleichen Vorwahl eintrifft, wird die gesendete Vorwahl ausgeblendet.
    - Telefonbuch: Hier kann die Datei angegeben werden, in der das lokale Telefonbuch zur Auflösung von Telefonnummer gespeichert wird. Existiert die Datei nicht, wird sie vom Programm angelegt.
    - Logdatei: Der Dateinamen, den man hier angeben kann, wird dazu benutzt, die Calls-Liste unter diesem Namen lokal auf dem Rechner zu speichern. Dieser Menüpunkt ist nur sichtbar, wenn die Config-Variable TELMOND\_LOG auf 'yes' gesetzt ist, dieses gilt auch für die eigentliche Anruf-Liste.

- Externes Suchprogramm benutzen: In diesem Bereich kann ein Programm angegeben werden, dass aufgerufen wird, wenn eine Telefonnummer mittels des lokalen Telefonbuches nicht aufgelöst werden kann. Nähere Infos sollten den entsprechenden Programmen beiliegen. Bis jetzt gibt es eine Anbindung an die Telefonbuch-CD KlickTel sowie von Marcel Wappler eine Anbindung an die Palm-Datenbank.
- Der Unterbereich Call Notification: Hier kann das bestimmt werden, ob ein Hinweis auf eingehende Telefonanrufe angezeigt werden soll und wie dieser sich optisch präsentiert.
  - Call Notification aktivieren: Bestimmt, ob Anrufe signalisiert werden sollen.
  - Call Notification anzeigen: Soll bei eingehenden Anrufen ein Hinweisfenster mit den Infos: angerufene MSN, Rufnummer des Anrufers und Datum/Uhrzeit erscheinen? Dafür ist es nötig, dass in der Datei config/isdn.txt die Variable OPT\_TELMOND auf ‘yes’ gesetzt wird.
    - \* Unterdrücken, wenn keine Nummer übertragen wurde: Soll Die Call Notification nicht angezeigt werden, wenn keine Rufnummer übertragen wurde.
    - \* Anzeigendauer: Diese Angabe beeinflusst die Dauer, wie lange das Call Notification-Fenster geöffnet bleiben soll. Die Angabe von “0” an dieser Stelle bewirkt, dass das Fenster nicht automatisch geschlossen wird.
    - \* Fontsize: Hiermit wird die Schriftgröße bestimmt. Dieses hat einen Einfluss auf die Größe des Fenster, da die notwendige Größe des Fenster anhand der tatsächlichen Größe der Mitteilung berechnet wird.
    - \* Farbe: Hiermit kann die Schriftfarbe ausgewählt werden. Ich selber benutzte die Farbe rot, damit ich es auch direkt wahrnehme.
- Der Unterbereich Phonebook: Die Seite Phonebook beinhaltet das Telefonbuch, welches zur Rufnummerrückmeldung der anrufenden Nummer als auch der eigenen MSN benutzt wird. Die Seite wird auch angezeigt, wenn die Konfigurationsvariable TELMOND\_LOG auf ‘no’ gesetzt ist, da die Rufnummerrückmeldung auch für die Anzeige des letzten Anrufes auf der Summary-Seite benutzt wird. Alternativ kann statt dem Telefonbuch auf dem Router auch eine lokale Datei ausgewählt werden.

Der Aufbau der Eintrag sieht wie folgt aus:

```
# Format:
# Telefonnummer=anzuweisender Name[, Wavefilename]
# 0241123456789=Testuser
00=unbekannt
508402=Fax
0241606*=Elsa AG Aachen
```

Dabei sind die ersten drei Zeilen Kommentare. Die vierte Zeile bewirkt, dass, wenn keine Rufnummer übermittelt wird, “unbekannt” angezeigt wird. In der fünften Zeile wird der MSN 508402 der Name “Fax” zugeordnet. Ansonsten ist das Format immer Telefonnummer=Name, der stattdessen angezeigt werden soll. In der sechsten Zeile ist noch die Möglichkeit demonstriert, eine Sammelrufnummer zu definieren. Damit wird erreicht, dass für alle Nebenstellen von 0241606 der Name angezeigt wird. Zu beachten hierbei ist, dass der erste Eintrag im Telefonbuch, welcher auf den Anruf passt, genommen

wird. Optional kann auch noch ein Wave-Datei angegeben werden, die abgespielt wird, wenn ein Telefonanruf von dieser Rufnummer eingeht.

Ab der Version 1.5.2 besteht jetzt auch die Möglichkeit auf der Seite Names das lokale Telefonbuch mit dem auf dem Router abgespeicherten (in `/etc/phonebook`) zu synchronisieren und umgekehrt. Dabei werden nicht nur einfach die Dateien ersetzt, sondern es werden die noch fehlende Einträge hinzugefügt. Gibt es eine Telefonnummer in beiden Telefonbüchern mit unterschiedlichen Namen, wird nachgefragt, welcher Eintrag genommen werden soll. Für die Synchronisierung des Telefonbuches auf dem Router ist noch anzumerken, dass dieses nur in der Ramdisk verändert wird, d.h. dass nach einem Reboot sämtliche Änderungen verloren gehen.

- Der Bereich Sound: Die Wave-Dateien, die hier angegeben werden, werden abgespielt, wenn das jeweilige Ereignis eingetreten ist.
  - E-Mail: Wenn der E-Mail-Checker auf einem angegebenen POP3-Server neue E-Mails vorfindet, wird die angegebene Wave-Datei abgespielt.
  - E-Mail-Error: Wenn ein Fehler beim Abrufen der E-Mails auftritt, wird diese Wave-Datei abgespielt.
  - Verbindung verloren: Wenn die Verbindung zum Router nicht mehr vorhanden ist (z.B. wenn der Router von einem anderen Client gerade neu gebootet wird), wird diese Wave-Datei abgespielt. Wenn die Option “Automatic Reconnect to router” nicht aktiviert ist, erscheint ausserdem eine MessageBox, die nachfragt, ob versucht werden soll, eine neue Verbindung zum Router aufzubauen.
  - Verbindungsmeldung: Wenn der Router eine Verbindung zum Internet aufgebaut hat, wird diese Wave-Datei abgespielt.
  - Verbindungsabbau: Wenn der Router die Verbindung zum Internet wieder abgebaut hat, wird diese Wave-Datei abgespielt.
  - Anrufmeldung: Wenn die Call Notification aktiviert ist und ein neuer Anruf eingeht, wird die angegebene Wave-Datei abgespielt.
  - Fax Notification: Die hier angegebene Wave-Datei wird nach dem Empfang neuer Faxe abgespielt.
- Der Bereich E-Mails
  - Accounts: Dieser Bereich dient dazu, die vorhandenen POP3-Accounts zu konfigurieren.
  - E-Mail-Check aktivieren: Soll der E-Mail-Checker automatisch nach neuen E-Mails suchen
    - \* Check jede x Min: Hiermit wird angegeben, wie oft der E-Mail-Checker automatisch nach neuen E-Mails suchen soll. Achtung: ein zu kurzes Intervall kann dazu führen, dass der Router komplett online bleibt! Dies ist der Fall, wenn das Intervall kleiner ist als der Hangup-Timeout des verwendeten Circuits.
    - \* TimeOut x Sec: Wie lange soll auf einen POP3-Server gewartet werden, bis er antwortet. Der Wert “0” bedeutet, dass kein TimeOut gesetzt wird.

- \* Auch wenn Router offline: Hiermit wird erreicht, dass der Router sich selbstständig einwählt, um nach E-Mails zu sehen. Nachdem alle POP3-Konten nach E-Mails überprüft worden sind, wird die Verbindung wieder getrennt. Um dieses Feature nutzen zu können, muss Dialmode auf 'auto' stehen. Achtung: Dadurch entstehen zusätzliche Kosten, wenn nicht gerade eine Flatrate benutzt wird!
  - \* Zu benutzender Circuit: Hiermit wird angegeben, welcher Circuit zur Einwahl beim E-Mail-Checken benutzt werden soll.
  - \* Anschliessend online bleiben: Soll direkt nach dem E-Mail-Check direkt die Verbindung getrennt werden oder eine Verbindungstrennung durch das Hangup-timeout realisiert werden.
  - \* E-Mail-Header laden: Sollen auch die E-Mail-Header geladen oder nur die Anzahl der vorhandenen E-Mails abgefragt werden? Das Laden der E-Mail-Header ist Voraussetzung, wenn man E-Mails direkt auf dem Server löschen möchte.
  - \* Notify only new E-Mails: Sollen nur neue E-Mails akustisch und mit dem Tray-Icon gemeldet werden
  - \* E-Mail-Client starten: Soll der angegebene E-Mail-Client automatisch gestartet werden, wenn neue E-Mails vorhanden sind.
  - \* E-Mail-Client: Hier wird der zu startende E-Mail-Client angegeben.
  - \* Param: Hier kann man zusätzliche Parameter angeben, die beim Start des E-Mail-Clients übergeben werden sollen. Wenn Outlook als E-Mail-Client benutzt wird (nicht Outlook Express), sollte /recyle als Parameter eingetragen werden, damit eine bereits geöffnete Instanz von Outlook beim Eintreffen von neuen E-Mails benutzt wird.
- Der Bereich Admin
    - root-Passwort: Hier sollte das Router-Passwort (in config/base.txt unter **PASSWORD** eingetragen) eingetragen werden, damit z.B. das Portforwarding lokal bearbeitet und wieder auf dem Router hinterlegt werden kann.
    - Dateien auf dem Router, die angezeigt werden sollen: Alle hier angegebenen Dateien, die sich auf dem Router befinden, können einfach per Maus-Click auf der Seite Admin/Dateien angezeigt werden. Somit kann man sich auf einfache Weise die Logfiles des Routers direkt im imonc anzeigen lassen.
    - Konfigdateien bearbeiten: Hier kann ausgewählt werden, ob die Konfigdateien alle im Editor geöffnet werden sollen (dies kann, wenn TXT-Dateien noch mit einem einfachen Editor verknüpft sind, dazu führen, dass sehr viele Instanzen des Editors geöffnet werden). Alternativ kann auch einfach nur das Verzeichnis geöffnet werden, so dass die Möglichkeit besteht, nur die Dateien auszuwählen, die bearbeitet werden sollen.
    - DynEisfaiLog: Wenn ein Account bei DynEisfair vorhanden ist, kann man hier seine Zugangsdaten eintragen und sich dann ein Log der Aktualisierung des Dienstes auf der Seite Admin/DynEisfairLog anschauen.
  - Der Bereich LaunchList dient dazu, die Launchliste zu konfigurieren. Diese wird nach einem erfolgreichen Connect ausgeführt, wenn die Option "Activate Launchlist" aktiviert ist.

- Programme: Alle hier eingetragenen Programme werden automatisch gestartet, sobald der Router eine Verbindung aufgebaut hat und die Launchliste aktiviert ist.
- LaunchList aktivieren: Soll die Launchliste beim erfolgreichen Verbindungsaufbau ausgeführt werden?
- Der Bereich Traffic dient dazu, dass TrafficInfo-Fenster den eigenen Bedürfnissen anzupassen. Von einem User habe ich den Hinweis bekommen, dass es mit älteren DirectX-Versionen offenbar Darstellungsfehler gibt.
  - Separates Traffic-Info-Fenster anzeigen: Soll eine grafische Kanalauslastung in einem separaten Fenster angezeigt werden? In dem Kontextmenü des Fensters kann man festlegen, ob das Fenster das Attribut StayOnTop bekommen soll. Dieses bewirkt, dass sich das Fenster immer über allen anderen Fenstern plaziert. Auch dieser Wert wird in der Registry abgespeichert und steht somit auch nach einem erneuten Programmstart wieder zur Verfügung.
  - Titelleiste anzeigen: Soll die Titelleiste des Traffic-Info-Fensters angezeigt werden? In der Titelzeile wird angezeigt, mit welchem Circuit der Router gerade online ist.
    - \* CPU-Auslastung in Titelleiste: Soll auch die CPU-Auslastung in der Titelzeile angezeigt werden?
    - \* Online-Zeit in Titelleiste: Soll die Onlinezeit des Kanals auch in der Titelzeile angezeigt werden?
  - Semi-transparentes Fenster: Soll das Fenster transparent dargestellt werden? Diese Funktion steht nur unter Windows 2000 und Windows XP zur Verfügung.
  - Farben: Hier werden die Farben für das TrafficInfo-Fenster definiert. Zu Berücksichtigen ist dabei, dass der DSL-Kanal und der erste ISDN-Kanal die selben Farbwerte zugewiesen bekommen.
  - Limits: Hier können die max. Übertragungswerte für DSL eingestellt werden – Upload und Download.
- Der Bereich Syslog dient dazu, die Anzeige der Syslog-Meldungen zu konfigurieren.
  - Syslog-Client aktivieren: Sollen Syslog-Meldungen im imonc angezeigt werden? Diese Option sollte ausgeschaltet sein, wenn ein externer Syslog-Client, wie zum Beispiel Kiwi's Syslog Client, benutzt wird.
  - Alle Meldungen ab Stufe anzeigen: Ab welcher Prioritätsstufe sollen die Syslog-Meldungen angezeigt werden? Es ist sinnvoll am Anfang mit der Stufe Debug anzufangen, um damit festzustellen, welche Meldungen einen interessieren. Anschliessend kann hier dann die entsprechende Stufe eingetragen werden.
  - Syslog-Meldungen in einer Datei speichern: Sollen die angezeigten Syslog-Meldungen in einer Datei gespeichert werden? In der Groupbox können dann die Meldungen ausgewählt werden, die in der Datei geloggt werden sollen. Für den Dateinamen sind folgende Platzhalter eingefügt worden:
    - %y** – wird durch das aktuelle Jahr ersetzt
    - %m** – wird durch den aktuellen Monat ersetzt
    - %d** – wird durch den aktuellen Tag ersetzt

- Portnamen anzeigen: Sollen statt den Portnummern deren Bedeutungen angezeigt werden?
- Firewall-Meldungen auch im User-Modus anzeigen: Hiermit wird festgelegt, dass Firewall-Meldungen auch im User-Modus angezeigt werden sollen.
- Der Bereich Fax dient dazu, die Faxanzeige vom imonc zu konfigurieren. Damit dieser Punkt angezeigt wird, muss mgetty bzw. faxrcv auf dem Router installiert sein (zu finden als OPT-Pakete auf der fli4l-Homepage).
  - Logdatei für Faxe: Den Dateinamen, den man hier angeben kann, wird dazu benutzt, die Fax-Liste unter diesem Namen lokal auf dem Rechner abzuspeichern.
  - Lokales Verzeichnis: Um die Faxe anzeigen zu können, müssen sie lokal gespeichert werden. Dieses kann hier eingestellt werden.
  - Aktualisierung: Es gibt zwei verschiedene Möglichkeiten, wie imonc mitbekommt, dass ein neues Fax eingegangen ist. Entweder wertet imonc die entsprechenden Syslogmeldungen aus (dazu muss natürlich der Syslog-Client im imonc aktiviert sein) oder er schaut regelmäßig selber in der Logdatei nach. Die erste Variante ist zu bevorzugen. Falls die zweite Variante genutzt wird, kann man noch angeben, wie oft die Faxübersichtsseite aktualisiert werden soll. Dabei ist zu beachten, dass dieser Wert keine Angabe in Sekunden ist, sondern noch mit der Angabe von Allgemein/Aktualisierungsintervall multipliziert wird.
- Der Bereich Grids dient dazu die Grids (Tabellen) im imonc an die eigenen Bedürfnisse anzupassen. Einerseits kann für jedes Grid angegeben werden, welche Spalten angezeigt werden sollen, andererseits gibt es die Möglichkeit für die Grids im Bereich Anrufe, Verbindungen und Faxe anzugeben, von wann ab die Infos angezeigt werden sollen.

### 7.2.5. Seite Anrufe

Die Seite Calls wird nur angezeigt, wenn die Konfigurationsvariable TELMOND\_LOG auf 'yes' eingestellt ist, denn sonst wird kein Anruf-Log geführt. Auf dieser Seite werden alle abgespeicherten Telefonanrufe angezeigt, die eingegangen sind, während der Router eingeschaltet war. Dabei kann umgeschaltet werden zwischen der Ansicht der lokal gespeicherten Anrufe oder nur der auf dem Router gespeicherten Anrufe. Wird bei der Anzeige der auf dem Router gespeicherten Anrufe der Zurücksetzen-Button gedrückt, wird das Logfile auf dem Router gelöscht.

In der Anruf-Übersicht kann mit der rechten Maustaste auf der Rufnummer oder der eigenen MSN diese ins Telefonbuch übernommen werden, um der Rufnummer bzw. MSN dort einen Namen zuzuweisen, der dann stattdessen angezeigt wird.

### 7.2.6. Seite Verbindungen

Neu ist ab der Version 1.4 die Anzeige der vom Router aufgebauten Verbindungen zum Internet. Diese befindet sich auf der Seite Connections. Somit hat man einen guten Überblick, wie sich der Router bei der automatischen Einwahl ins Internet verhält. Damit diese Seite angezeigt werden kann, muss in der Datei config/base.txt die Variable IMOND\_LOG auf 'YES' gesetzt werden.

Genauso wie bei der Anruf-Übersicht kann auch hier zwischen den lokal gespeicherten und auf dem Router gespeicherten Verbindungen umgeschaltet werden. In der Ansicht der auf dem

Router gespeicherten Verbindungen bewirkt ein Drücken des Zurücksetzen-Buttons, dass das Logfile auf dem Router gelöscht wird.

Angezeigt werden pro Verbindung

- Provider
- Startdatum und -zeit
- Enddatum und -zeit
- Onlinezeit
- Abrechnungszeit
- entstandene Kosten
- empfangene Zeichen
- gesendete Zeichen

### 7.2.7. Seite Fax

Damit die Seite Faxe angezeigt wird, muss auf dem Router entweder das `OPT_MGETTY` von Michael Heimbach oder `OPT_MGETTY` von Felix Eckhofer installiert werden. Diese gibt es auf der fli4l-Homepage unter OPT-Pakete. Auf dieser Seite werden dann alle eingegangenen Faxe aufgelistet. Das Kontextmenü der Übersicht bietet mehrere Möglichkeiten, diese stehen allerdings nur im Admin-Modus zur Verfügung:

- Es kann ein Fax angezeigt werden. Dazu muss unter Admin/Remoteupdate der Pfad für das fli4l-Verzeichnis korrekt gesetzt werden, da die Faxe auf dem Router in gepackter Form vorliegen und somit gzip aus dem fli4l-Paket benötigt wird. Alternativ kann gzip.exe und win32gnu.dll auch ins imonc-Verzeichnis kopiert werden. Kann gzip.exe nicht an einer der beiden Stellen gefunden werden, wird stattdessen der Webserver des Routers probiert zu öffnen (direkt mit dem Aufruf des richtigen CGIs).
- Ein einzelnes Fax kann gelöscht werden. Dabei wird das Fax sowohl lokal als auch auf dem Router gelöscht (sowohl die eigentliche Faxdatei, als auch der Eintrag in den Logdateien).
- Sämtliche auf dem Router befindlichen Faxe löschen. Damit werden die Faxe und die Logdatei auf dem Router gelöscht. Die Faxe werden nicht aus der lokalen Logdatei gelöscht.

Genauso wie bei der Anruf-Übersicht kann auch hier zwischen den lokal gespeicherten und auf dem Router gespeicherten Faxen umgeschaltet werden.

### 7.2.8. Seite E-Mail

Diese Seite wird nur angezeigt, wenn im Config-Dialog mindestens ein aktiviertes POP3-E-Mail-Konto eingerichtet worden ist.

Die Seite E-Mail dürfte sich eigentlich selber erklären. Hiermit wird der mittlerweile eingebaute E-Mail-Checker beobachtet. Ist die Option "Check even if the router is offline" nicht



aktiviert, überprüft der E-Mail-Checker alle E-Mail-Konten nach E-Mails, sobald der Router online ist und anschließend im eingestellten Intervall. Ist die genannte Option aktiviert, überprüft der E-Mail-Checker im eingestellten Intervall. Ist der Router gerade online, wird die bestehende Verbindung benutzt. Ist er nicht online, wird eine Verbindung selbständig mit dem ausgewählten Circuit hergestellt, die, sobald alle E-Mail-Konten abgearbeitet sind, wieder getrennt wird. Damit man diese Option nutzen kann, muss Dialmode auf "auto" stehen.

Sind E-Mails auf dem POP3-Server vorhanden, wird entweder automatisch der eingestellte E-Mail-Client gestartet oder ein neues Symbol im Tray neben der Uhr angezeigt, welches als Hint die Anzahl der E-Mails auf jedem Server liefert. Ein Doppelclick startet dann den eingestellten E-Mail-Client. Ist ein Fehler bei einem der E-Mail-Konten aufgetreten, erscheint einerseits ein Hinweis in der E-Mail-History, andererseits wird das E-Mail-TrayIcon angezeigt, welches dadurch gekennzeichnet ist, dass die obere rechte Ecke rot gefärbt ist.

In der E-Mail-Übersicht kann man mit dem Kontextmenü Mails direkt auf dem Server löschen, ohne sie vorher komplett downloaden zu müssen. Dies geschieht, indem mit der rechten Maustaste das Kontextmenü angezeigt wird. Dabei sollte eine Zelle der entsprechenden Zeile markiert sein, wo die zu löschende E-Mail eingetragen ist. Im Kontextmenü wählt man den einzige Punkt Delete MailMessage aus.

### 7.2.9. Admin

Dieser Abschnitt steht nur zur Verfügung, wenn sich imonc im Admin-Modus befindet.

Der erste Punkt liefert eine Übersicht über die verwendeten Circuits – sprich Internetprovider – die der Router automatisch per LCR auswählt. Ein Doppelclick auf einen Provider in der Providerübersicht zeigt an, für welche Zeiträume der Circuit in config/base.txt definiert worden ist.

Der zweite Punkt dort ist die Möglichkeit ein Fernupdate auf dem Router einzuspielen. Dabei kann ausgewählt werden, welche fünf Programmpakete (Kernel, Rootfilesystem, Opt-Datei, rc.cfg und syslinux.cfg) auf den Router kopiert werden sollen. Damit man das Update einspielen kann, muss man zuerst mal das fli4l-Verzeichnis angeben, damit imonc weiss, woher es die nötigen Dateien nehmen soll. Ausserdem muss angegeben werden, in welchem Unterverzeichnis die Konfigurationsdateien liegen (standardmäßig config), um die Opt-Datei, rc.cfg und syslinux.cfg jeweils neu zu erzeugen. Es ist ratsam, einen Reboot nach dem Einspielen des Updates durchführen, damit die Änderungen auch direkt wirksam werden. Wird während des Updates nach einem Passwort nachgefragt, ist das Passwort gemeint, welches in config/base.txt unter PASSWORD eingetragen ist.

Um die Beschränkung des Port-Forwarding zu umgehen, dass ein Port nur an genau einen Client-Rechner gebunden ist, besteht jetzt die Möglichkeit, die Konfiguration auf dem Router zu editieren. Damit die Änderungen aktiv werden, muss die Verbindung neu hergestellt werden. Da die Datei nur in der Ramdisk ersetzt wird, bleiben die Änderungen nur bis zum nächsten Neustart des Routers erhalten. Um die Änderungen dauerhaft zu speichern, muss ein neues Opt-File auf dem Router installiert werden mit einer geeignet angepassten base.txt aus dem Config-Verzeichnis.

Der vierte Punkt auf der Admin-Seite – Dateien – dient dazu, Konfigurations- und Logdateien des Routers einfach per Maus-Click anzuzeigen. Die Auswahlliste wird über den Punkt Config/Admin und dort "files on router to view" konfiguriert. Anschliessend kann einfach über die ComboBox auf dieser Seite ausgewählt werden, welche Datei angezeigt werden soll.

Der fünfte Punkt ist die Seite DynEisfairLog, sie erscheint nur wenn im Config-Dialog unter

Admin die Zugangsdaten des DynEisfair-Accounts eingetragen worden sind. Ist dies geschehen, wird auf dieser Seite Log des Dienstes angezeigt.

Als letzten Punkt gibt es noch die Seite Hosts. Hier werden alle in der Datei `/etc/hosts` eingetragenen Rechner angezeigt. Weiterhin wird probiert jeden dieser dort eingetragenen Rechner anzupingen und das Ergebnis davon wird ebenfalls angezeigt. Somit kann man schnell rausbekommen, welche dieser Rechner eingeschaltet sind.

### 7.2.10. Seiten Fehler, Syslog und Firewall

Die Seiten Fehler, Syslog und Firewall werden nur angezeigt, wenn in den entsprechenden Logs Einträge vorhanden sind. Die Einträge der Seiten Syslog und Firewall werden nur angezeigt, wenn man im Admin-Modus ist.

Auf der Seite Fehler werden sämtliche imonc/imond-spezifischen Fehler festgehalten. Wenn Probleme bestehen, kann unter Umständen ein Blick auf diese Seite die Ursache der Probleme anzeigen.

Auf der Seite Syslog werden die ankommende Syslog-Meldungen angezeigt, bis auf die Meldungen der Firewall. Diese werden auf der eigenen Seite Firewall dargestellt. Damit dies funktioniert, muss die Variable `OPT_SYSLOGD` in der Konfigurationsdatei `config/base.txt` auf 'yes' gesetzt werden. Ausserdem muss die Variable `SYSLOGD_DEST` auf die IP des Clients gesetzt werden (genau: `SYSLOGD_DEST='@100.100.100.100` – wobei die IP natürlich an die IP des Clients angepasst werden muss). Angezeigt wird neben der eigentlichen Syslog-Meldung auch Datum, Uhrzeit, IP des Senders und die Prioritätsstufe.

Damit die Firewall-Meldungen bei den ganzen Syslog-Meldungen nicht untergehen, werden diese auf der separaten Seite Firewall angezeigt. Damit die Firewall-Meldungen angezeigt werden können, muss zusätzlich in der Datei `config/base.txt` die Konfigurationsvariable `OPT_KLOGD` auf 'yes' gesetzt werden.

### 7.2.11. Seite News

Auf dieser Seite werden, vorausgesetzt die Option ist im Config-Bereich des Imonc aktiviert, die News, welche auf der fli4l-Homepage angezeigt werden, auch direkt im Imonc angezeigt werden. Dazu wird mittels des http-Protokolls die URL `http://www.fli4l.de/german/news.xml` abgerufen. Neben den News werden mittlerweile auch die fünf aktuellsten Opt-Pakete angezeigt. Dazu wird die URL `http://www.fli4l.de/german/imonc_opt_show.php` abgefragt. Außerdem wird in der Statusleiste vom Imonc die Überschriften der News alternierend angezeigt.

## 7.3. Unix/Linux-Client imonc

Für Linux gibt es mittlerweile 2 Versionen: eine textbasierte (imonc) und eine mit graphischer Oberfläche (ximonc). Den Source zu ximonc findet man im Verzeichnis `src`. Die Dokumentation für ximonc wird erst in der 1.5-Final-Version zur Verfügung stehen. Ein erfahrener Linux-User sollte aber mit den Sources keine Probleme haben.

Beschränken wir uns daher hier zunächst auf die textbasierte Version von imonc: Dieses ist ein curses-basiertes Programm, hat also keine graphische Oberfläche. Der Source liegt im Verzeichnis `unix`.

Installation:

## 7. Client-/Server-Schnittstelle imon

```
cd unix
make install
```

imonc wird dabei in /usr/local/bin installiert.  
Aufruf:

```
imonc hostname
```

Dabei ist als hostname der Name oder die IP-Adresse des fli4l-Routers anzugeben, also z.B.

```
imonc fli4l
```

imonc zeigt folgende Informationen:

- Datum/Uhrzeit des fli4l-Routers
- Momentan eingestellte Route
- Default-Route-Circuits
- ISDN-Kanäle

**Status** : Calling/Online/Offline

**Name** : Telefonnummer des Gegners oder Circuit-Name

**Time** : Online-Zeit

**Charge-Time** : Online-Zeit unter Berücksichtigung des Zeittaktes

**Charge** : Berechnete Kosten

Mögliche Kommandos sind:

Nr	Befehl	Bedeutung
0	quit	Programm beenden
1	enable	Aktivieren
2	disable	Deaktivieren
3	dial	Wählen
4	hangup	Einhängen
5	reboot	Neu booten
6	timetable	Zeittabelle ausgeben
7	dflt route	Neuen Default-Route-Circuit bestimmen
8	add channel	2. Kanal hinzuschalten
9	rem channel	2. Kanal deaktivieren

Zu den Kommandos im Einzelnen:

**0 – quit** Die Verbindung zum imond-Server wird abgebaut und das Programm beendet.

**1 – enable** Alle Circuits werden auf dialmode “auto” gestellt. Das ist auch der Default-Zustand von fli4l nach dem Booten. Das heisst, dass fli4l bei einem Verbindungsaufbauwunsch eines Rechners im Netz automatisch rauswählt.

**2 – disable** Alle Circuits werden auf dialmode “off” gestellt. Damit ist fli4l so gut wie tot, bis er mit dem Enable-Kommando wieder geweckt wird.

- 3 – dial** Manuelle Wahl auf dem Default-Route-Circuit. Ist eher für Testzwecke gedacht, da fli4l normalerweise automatisch wählt.
- 4 – hangup** Manuelles Einhängen: damit kann man dem automatischen Einhängen von fli4l zuvorkommen.
- 5 – reboot** fli4l wird neu gebootet. Ziemlich überflüssiges Kommando ...
- 6 – timetable** Es wird die Zeittabelle für die Default-Route-Circuits ausgegeben. Beispiel: s.o.
- 7 – default route circuit** Manuelles Wechseln des Default-Route-Circuits. Kann z.B. dann sinnvoll sein, wenn man das automatische LC-Routing von fli4l für eine Weile ausser Kraft setzen will, da einige Provider einen Zugriff auf das eigene Postfach nur über den eigenen Internet-Zugang erlauben.
- 8 – add channel** Hier kann der 2. ISDN-Kanal manuell hinzugeschaltet werden. Voraussetzung: ISDN\_CIRC\_x\_BUNDLING ist 'yes'.
- 9 – remove channel** Abschalten des 2. ISDN-Kanals. Siehe auch "add channel".

Sonst gelten bei diesen Kommandos dieselben Bemerkungen wie für den Windows-imond-Client `imonc.exe`.

Noch zu bemerken ist: Ab fli4l-1.4 ist es nun auch möglich, auf dem fli4l-Router selbst einen "minimalisierten" imon-Client zu installieren, nämlich durch Setzen von `OPT_IMONC='yes'` im Paket `TOOLS`.

Damit kann man nun auch an der fli4l-Konsole bestimmte Einstellungen, z.B. Routing etc. mit `imonc` vornehmen. Achtung: Dieser Mini-`imonc` funktioniert nur auf dem fli4l-Router selbst! Auf einem Linux-/Unix- Client ist immer der "große Bruder" `unix/imonc` zu verwenden.

# A. Anhang zum Basispaket

## A.1. Nullmodemkabel

Für die Verwendung des optionalen Programmpakets PPP (Seite ??) benötigt man ein Nullmodemkabel.

Dieses muss mindestens drei Adern haben. Hier die Anschluss-Belegung:



Bei den Steckern müssen die im Schaltbild gezeigte Brücken eingelötet werden.

## A.2. Serielle Konsole

fli4l kann ohne Monitor und Tastatur eingesetzt werden. Ein Nachteil davon ist, dass eventuelle Fehlermeldungen nicht bemerkt werden, weil sich nicht alle Meldungen über die syslog-Schnittstelle umleiten lassen.

Eine Möglichkeit ist die Umlenkung der Konsole-Meldungen auf seinen PC oder auf ein klassisches Terminal, nämlich über die serielle Schnittstelle. Die Konfiguration erfolgt über die Variablen [SER\\_CONSOLE](#) (Seite 30), [SER\\_CONSOLE\\_IF](#) (Seite 31) und [SER\\_CONSOLE\\_RATE](#) (Seite 31).

Rechner mit älteren Mainboards/Karten unterstützen keine höheren Geschwindigkeiten als 38400 Baud. Deshalb sollte man es zunächst mit höchstens 38400 Baud probieren, bevor man sich an höhere Geschwindigkeiten heranwagt. Da lediglich Text-Ausgaben über die Konsole gehen, sind höhere Geschwindigkeiten eigentlich auch gar nicht notwendig.

Sämtliche Meldungen, die normalerweise auf der Konsole ausgegeben werden, werden nun auf die serielle Schnittstelle gelenkt – auch die Meldungen des Bootvorgangs!

Als Kabel zum Terminal oder PC mit Terminalemulation kommt ein [Nullmodemkabel](#) (Seite 141) zum Einsatz. Wir raten aber davon ab, ein Standard-Nullmodemkabel zu verwenden, weil dort normalerweise auch die Handshake-Leitungen verdrahtet sind. Ist das Terminal bzw. der PC abgeschaltet (oder die Terminalemulation nicht geladen), kann es bei Verwendung eines Standard-Nullmodemkabels zu einem Einfrieren von fli4l kommen!

Deshalb ist hier eine spezielle Verdrahtung notwendig, um fli4l auch mit abgeschaltetem Terminal betreiben zu können. Es wird dafür ein dreiadriges Kabel benötigt, wobei einige Kontakte an den Steckern überbrückt werden müssen. Siehe dazu bei [Nullmodemkabel](#) (Seite 141).

### A.3. Programme

Um Platz auf dem Bootmedium zu sparen, wird unter anderem das Paket “BusyBox” verwendet. Das Programm ist ein einzelnes Executable, welches die Standard-Unix-Programme

```
[, [[, arping, ash, base64, basename, bbconfig, blkid, bunzip2, 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, lsmmod, lzcat, makedevs, md5sum, mdev, mkdir, mknod,
mkswap, modprobe, mount, mv, nameif, nice, nslookup, ping, ping6, poweroff,
ps, pscan, pwd, reboot, reset, rm, rmmmod, 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
```

nachbildet. Zumeist sind es jedoch “minimalistische” Implementationen, welche nicht den vollen Funktionsumfang abdecken, aber voll auf den bescheidenen Anforderungen von fli4l genügen.

BusyBox steht unter GPL und ist als Source komplett erhältlich unter

<http://www.busybox.net/>

### A.4. Andere i4l-Tools

Es gibt für isdn4linux viele weitere Tools, die auch fli4l bereichern würden. Das Problem ist leider der Platz! Bestimmt wäre isdnlog als Tool zum Berechnen der Online-Gebühren wesentlich geeigneter, jedoch ist isdnlog einfach zu fett!

imond braucht weniger als 10% des Platzes, übernimmt dabei Monitoring, Controlling und LC-Routing, wenn auch nicht alles ganz perfekt ist.

### A.5. Fehlersuche

Hilfreich bei der Fehlersuche sind natürlich einmal die Konsole-Outputs. Diese rauschen aber oft einfach so durch, dass man gar nicht mehr mitlesen kann. Hinweis: Mit SHIFT-BILD-RAUF kann man zurück-, mit SHIFT-BILD-RUNTER wieder vorblättern.

Falls im Betrieb des Routers Fehlermeldungen “try-to-free-pages” auftreten ist zuwenig für Programme nutzbares RAM übrig. Als Abhilfe stehen dann folgende Optionen zur Verfügung:

- mehr RAM einbauen
- weniger Opt-Pakete einsetzen
- eine Festplatteninstallation nach [Typ B](#) (Seite 12) durchführen

Auch proc-Dateien können bei der Fehlersuche helfen. z.B. gibt der Befehl

```
cat /proc/interrupts
```

die von den Treibern verwendeten Interrupts aus – nicht die tatsächlich von der Hardware belegten!

Weitere interessante Dateien unter /proc sind devices, dma, ioports, kmsg, meminfo, modules, uptime, version und pci (falls der Router einen PCI-Bus hat).

Meist liegt ein Verbindungsproblem bei ipppd vor, insbesondere bei der Authentifizierung. Dann helfen oft die Variablen

```
OPT_SYSLOGD='yes'
```

```
OPT_KLOGD='yes'
```

in config/base.txt und

```
ISDN_CIRC_x_DEBUG='yes'
```

in config/isdn.txt weiter.

## A.6. Literaturhinweise

- 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. Präfixe

Bei den Einheiten richten die Präfixe in dieser Doku sich nach IEC 60027-2.

Siehe: <http://physics.nist.gov/cuu/Units/binary.html>.

## A.8. Gewähr und Haftung

Natürlich kann für das gesamte fli4l-Paket oder für Teile davon keine Gewähr dafür übernommen werden, dass es überhaupt funktioniert oder dass irgendeine Dokumentation in diesem Verzeichnis oder einem der Unterverzeichnisse korrekt ist.

Auch ist jede Haftung wegen evtl. entstandender Schäden oder Kosten ausgeschlossen!

## A.9. Danke

In diesem Abschnitt der Dokumentation soll all denen durch namentliche Nennung gedankt werden, die zur Entwicklung von fli4l beitragen bzw. beigetragen haben.

### A.9.1. Projektgründung

Meyer, Frank

Frank startete am 04.05.2000 das Projekt fli4l!

Siehe: <http://www.fli4l.de/home/eigenschaften/historie/>

### A.9.2. Entwickler- und Testteam

**Das fli4l-Team bilden (in alphabetischer Reihenfolge):**

Charrier, Bernard (*französische Übersetzung*)  
Eckhofer, Felix (*Dokumentation, Howtos*)  
Franke, Roland (*OW, FBR*)  
Hilbrecht, Claas (*VPN, Kernel*)  
Klein, Sebastian (*Kernel, Wlan*)  
Knipping, Michael (*Accounting*)  
Krister, Stefan (*Opt-Cop, lcd4linux*)  
Miksch, Gernot (*LCD*)  
Schiefer, Peter (*fli4l-CD, Opt-Cop, Webseite, Releasemanagement*)  
Schliesing, Manfred (*Tester*)  
Schulz, Christoph (*FBR, IPv6, Kernel*)  
Siebmanns, Harvey (*Dokumentation, englische Übersetzung*)  
Spieß, Carsten (*Dsltool, Hwsupp, Rrdtool, Webgui*)  
Vosselman, Arwin (*LZS-Kompression, Dokumentation*)  
Weiler, Manuela (*CD-Versand, Kassenwart*)  
Weiler, Marcel (*Qualitätsmanagement*)  
Wolters, Florian (*Firmware, Kernel*)



### A.9.3. Entwickler- und Testteam (nicht mehr aktive)

Arndt, Kai-Christian (*USB*)  
Bauer, Jürgen (*LCD-Package, fliwiz*)  
Behrends, Arno (*Support*)  
Blokland, Kees (*Englische Übersetzung*)  
Bork, Thomas (*lpdsrv*)  
Bußmann, Lars (*Tester*)  
Cerny, Carsten (*Webseite, fliwiz*)  
Dawid, Oliver (*dhcp, uClibc*)  
Ebner, Hannes (*QoS*)  
Fischer, Joerg (*Tester*)  
Frauenhoff, Peter (*Tester*)  
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*)  
Schmitts, Jupp (*Tester*)  
Strigler, Stefan (*GTK-Imonc, Opt-DB, NG*)  
Wallmeier, Nico (*Windows-Imonc*)  
Walter, Gerd (*UMTS*)  
Walter, Oliver (*QoS*)  
Wolter, Jean (*Paketfilter, uClibc*)  
Zierer, Florian (*Wunschliste*)

#### **A.9.4. Sponsoren**

Auch ist mittlerweile fli4l als Wort-/Bildmarke eingetragen. Folgende fli4l-Anwender (neben einigen, die nicht genannt werden wollten) haben geholfen das dafür nötige Geld zusammen zu bekommen:

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

Seit einiger Zeit hat fli4l nun auch seine eigenen Sponsoren, die mit Ihrer (Hardware-)Spende die Weiterentwicklung von fli4l unterstützen. Dabei handelt es sich um Adapter, CompactFlash und Ethernetkarten.

Hardwarespender (in alphabetischer Reihenfolge):

Baglatzis, Stephanos  
Bauer, Jürgen  
Dross, Heiko  
Kappenhagen, Wenzel  
Kipka, Joachim  
Klopfer, Tom  
Peiser, Steffen  
Reichelt, Detlef  
Reinard, Louis  
Stärkel, Christopher

Weitere Sponsoren sind auf der fli4l-Homepage gelistet:

<http://www.fli4l.de/sonstiges/sponsoren/>

## A.10. Feedback

Kritik, Feedback und Zusammenarbeit ist jederzeit willkommen.

Die primäre Anlaufstelle dafür sind die fli4l-Newsgroups. Wer Probleme bei der Einrichtung eines fli4l-Routers hat, sollte sich erst FAQ, Howtos und NG-Archiv anschauen, bevor er sich an die Newsgroups wendet. Informationen über die verschiedenen Gruppen und die Netiquette findet man auf der fli4l-Webseite:

<http://www.fli4l.de/hilfe/newsgruppen/>

<http://www.fli4l.de/hilfe/faq/>

<http://www.fli4l.de/hilfe/howtos/>

Gerade weil für fli4l-Router meist ältere Hardware eingesetzt wird, kann es immer wieder mal zu Problemen kommen. Informationen können anderen fli4l-Usern bei Problemen mit der Hardware weiterhelfen, denn es gibt immer wieder Probleme mit den PC-Karten bzgl. I/O-Adressen, Interrupts und so weiter.

Auf der fli4l-Webseite wurde eine Netzwerkkarten-Datenbank eingerichtet, in die man z.B. die passenden Treiber für eine bestimmte Karte eintragen kann.

<http://www.fli4l.de/hilfe/nic-db/>

Viel Spaß mit fli4l!

# Abbildungsverzeichnis

3.1. Struktur des Paketfilters . . . . .	45
3.2. Verzeichnisstruktur fli4l . . . . .	54
5.1. Einstellungen . . . . .	111
5.2. Einstellungen für Remoteupdate . . . . .	112
5.3. Einstellungen für HD-pre-install . . . . .	113

# Tabellenverzeichnis

3.1. Übersicht über die (Zusatz-)Pakete . . . . .	14
3.2. Automatische Einstellung der maximalen Verbindungsanzahl . . . . .	29
3.3. Typen von Netzwerkpräfixen . . . . .	42
3.4. Aktionen in Paketfilterregeln . . . . .	46
3.5. Quell- und Zieleinschränkungen in Paketfilterregeln . . . . .	47
3.6. Zustandseinschränkungen in Paketfilterregeln . . . . .	50
3.7. Im Lieferumfang von fli4l enthaltene Schablonen . . . . .	53
3.8. Verfügbare Conntrack-Helfer im Paketfilter . . . . .	71
3.9. Format der Imond-Logdatei . . . . .	80
3.10. Verfügbare Circuit-Typen . . . . .	82
3.11. Circuit-Zustände . . . . .	89
3.12. Circuit-Zustandsübergänge . . . . .	90
3.13. Verfügbare Wählmodi . . . . .	91
3.14. fli4lctrl-Befehle . . . . .	94
3.15. fli4lctrl-Statusbefehle . . . . .	95

# Index

base.txt, [14](#)  
BEEP, [30](#)  
Beispiel-Datei(base.txt), [14](#)  
BOOT\_TYPE, [24](#)  
BOOTMENU\_TIME, [25](#)  
BUILDDIR, [114](#)  
  
CIRC\_CLASS\_N, [92](#)  
CIRC\_CLASS\_x\_NAME, [92](#)  
CIRC\_N, [82](#)  
CIRC\_ONLINE, [97](#)  
CIRC\_x\_BUNDLE, [88](#)  
CIRC\_x\_CHARGEINT, [86](#)  
CIRC\_x\_CLASS\_N, [87](#)  
CIRC\_x\_CLASS\_y, [87](#)  
CIRC\_x\_DEBUG, [87](#)  
CIRC\_x\_DEPS, [87](#)  
CIRC\_x\_DIALMODE, [83](#)  
CIRC\_x\_ENABLED, [83](#)  
CIRC\_x\_HUP\_TIMEOUT, [86](#)  
CIRC\_x\_NAME, [82](#)  
CIRC\_x\_NETS\_IPV4\_N, [83](#)  
CIRC\_x\_NETS\_IPV4\_y, [83](#)  
CIRC\_x\_NETS\_IPV6\_N, [83](#)  
CIRC\_x\_NETS\_IPV6\_y, [83](#)  
CIRC\_x\_PRIORITY, [84](#)  
CIRC\_x\_PROTOCOLS, [84](#)  
CIRC\_x\_ROUTE\_DEV, [98](#)  
CIRC\_x\_ROUTE\_GATEWAY\_IPV4, [98](#)  
CIRC\_x\_ROUTE\_GATEWAY\_IPV6, [99](#)  
CIRC\_x\_TIMES, [85](#)  
CIRC\_x\_TYPE, [82](#)  
CIRC\_x\_UP, [84](#)  
CIRC\_x\_USEPEERDNS, [86](#)  
CIRC\_x\_WAIT, [87](#)  
COMP\_TYPE\_OPT, [28](#)  
COMP\_TYPE\_ROOTFS, [27](#)  
CONSOLE\_BLANK\_TIME, [30](#)  
  
DEBUG\_ENABLE\_CORE, [32](#)  
DEBUG\_IP, [32](#)  
DEBUG\_IPTABLES, [32](#)  
DEBUG\_MDEV, [32](#)  
DEBUG\_MODULES, [31](#)  
DEBUG\_STARTUP, [31](#)  
DIALMODE, [91](#)  
DNS\_FORWARDERS, [78](#)  
DOMAIN\_NAME, [77](#)  
  
FILESONLY, [114](#)  
FLI4L\_UUID, [28](#)  
ftp, [71](#)  
  
h323, [71](#)  
HOSTNAME, [23](#)  
HOSTNAME\_ALIAS\_N, [78](#)  
HOSTNAME\_ALIAS\_x, [78](#)  
HOSTNAME\_IP, [78](#)  
HOSTNAME\_IP6, [40](#)  
  
IMOND\_ADMIN\_PASS, [79](#)  
IMOND\_BEEP, [80](#)  
IMOND\_DIAL, [80](#)  
IMOND\_ENABLE, [80](#)  
IMOND\_LED, [79](#)  
IMOND\_LOG, [80](#)  
IMOND\_LOGDIR, [80](#)  
IMOND\_PASS, [79](#)  
IMOND\_PORT, [79](#)  
IMOND\_REBOOT, [80](#)  
IMOND\_ROUTE, [80](#)  
IP\_CONNTRACK\_MAX, [28](#)  
IP\_DYN\_ADDR, [98](#)  
IP\_NET\_N, [36](#)  
IP\_NET\_x, [36](#)  
IP\_NET\_x\_COMMENT, [38](#)  
IP\_NET\_x\_DEV, [37](#)  
IP\_NET\_x\_MAC, [37](#)

- IP\_NET\_x\_NAME, 38
- IP\_NET\_x\_TYPE, 38
- IP\_ROUTE\_N, 43
- IP\_ROUTE\_x, 43
- IPV6\_NET\_N, 40
- IPV6\_NET\_x, 40
- IPV6\_NET\_x\_ADVERTISE, 41
- IPV6\_NET\_x\_ADVERTISE\_DNS, 41
- IPV6\_NET\_x\_DEV, 40
- IPV6\_NET\_x\_NAME, 41
- IPV6\_ROUTE\_N, 44
- IPV6\_ROUTE\_x, 44
- irc, 71
  
- KERNEL\_BOOT\_OPTION, 27
- KERNEL\_VERSION, 27
- KEYBOARD\_LOCALE, 33
  
- LIBATA\_DMA, 25
- LOCALE, 30
- LOGIP\_LOGDIR, 102
  
- Masquerading, 69
- MKFLI4L\_DEBUG\_OPTION, 115
- MOUNT\_BOOT, 25
  
- NET\_DRV\_N, 33
- NET\_DRV\_x, 34
- NET\_DRV\_x\_OPTION, 34
- NET\_PREFIX\_x, 41
- NET\_PREFIX\_x\_NAME, 42
- NET\_PREFIX\_x\_STATIC\_IPV4, 42
- NET\_PREFIX\_x\_STATIC\_IPV6, 42
- NET\_PREFIX\_x\_TYPE, 42
- NET\_PREFIX\_x\_ULA\_DEV, 43
  
- OPT\_CIRCUIT\_STATUS, 98
- OPT\_HOTPLUG\_PCI, 105
- OPT\_IMOND, 78
- OPT\_IPV4, 36
- OPT\_IPV6, 40
- OPT\_KLOGD, 102
- OPT\_LOGIP, 102
- OPT\_MAKEKBL, 33
- OPT\_NET\_PREFIX, 41
- OPT\_PNP, 103
- OPT\_SYSLOGD, 100
- OPT\_Y2K, 102
  
- PASSWORD, 23
- PF6\_FORWARD\_ACCEPT\_DEF, 73
- PF6\_FORWARD\_LOG, 74
- PF6\_FORWARD\_LOG\_LIMIT, 74
- PF6\_FORWARD\_N, 74
- PF6\_FORWARD\_POLICY, 73
- PF6\_FORWARD\_REJ\_LIMIT, 74
- PF6\_FORWARD\_UDP\_REJ\_LIMIT, 74
- PF6\_FORWARD\_x, 74
- PF6\_FORWARD\_x\_COMMENT, 75
- PF6\_INPUT\_ACCEPT\_DEF, 72
- PF6\_INPUT\_ICMP\_ECHO\_REQ\_LIMIT, 72
- PF6\_INPUT\_ICMP\_ECHO\_REQ\_SIZE, 73
- PF6\_INPUT\_LOG, 72
- PF6\_INPUT\_LOG\_LIMIT, 72
- PF6\_INPUT\_N, 73
- PF6\_INPUT\_POLICY, 72
- PF6\_INPUT\_REJ\_LIMIT, 72
- PF6\_INPUT\_UDP\_REJ\_LIMIT, 72
- PF6\_INPUT\_x, 73
- PF6\_INPUT\_x\_COMMENT, 73
- PF6\_LOG\_LEVEL, 71
- PF6\_OUTPUT\_ACCEPT\_DEF, 75
- PF6\_OUTPUT\_LOG, 75
- PF6\_OUTPUT\_LOG\_LIMIT, 75
- PF6\_OUTPUT\_N, 75
- PF6\_OUTPUT\_POLICY, 75
- PF6\_OUTPUT\_REJ\_LIMIT, 75
- PF6\_OUTPUT\_UDP\_REJ\_LIMIT, 75
- PF6\_OUTPUT\_x, 76
- PF6\_OUTPUT\_x\_COMMENT, 76
- PF6\_POSTROUTING\_N, 77
- PF6\_POSTROUTING\_x, 77
- PF6\_POSTROUTING\_x\_COMMENT, 77
- PF6\_PREROUTING\_N, 77
- PF6\_PREROUTING\_x, 77
- PF6\_PREROUTING\_x\_COMMENT, 77
- PF6\_USR\_CHAIN\_N, 76
- PF6\_USR\_CHAIN\_x\_NAME, 76
- PF6\_USR\_CHAIN\_x\_RULE\_N, 76
- PF6\_USR\_CHAIN\_x\_RULE\_x, 76
- PF6\_USR\_CHAIN\_x\_RULE\_x\_COMMENT, 77
- PF\_FORWARD\_ACCEPT\_DEF, 57
- PF\_FORWARD\_LOG, 57

- PF\_FORWARD\_LOG\_LIMIT, [57](#)
- PF\_FORWARD\_N, [58](#)
- PF\_FORWARD\_POLICY, [57](#)
- PF\_FORWARD\_REJ\_LIMIT, [57](#)
- PF\_FORWARD\_UDP\_REJ\_LIMIT, [57](#)
- PF\_FORWARD\_x, [58](#)
- PF\_FORWARD\_x\_COMMENT, [58](#)
- PF\_INPUT\_ACCEPT\_DEF, [55](#)
- PF\_INPUT\_ICMP\_ECHO\_REQ\_LIMIT, [56](#)
- PF\_INPUT\_ICMP\_ECHO\_REQ\_SIZE, [56](#)
- PF\_INPUT\_LOG, [56](#)
- PF\_INPUT\_LOG\_LIMIT, [56](#)
- PF\_INPUT\_N, [56](#)
- PF\_INPUT\_POLICY, [55](#)
- PF\_INPUT\_REJ\_LIMIT, [56](#)
- PF\_INPUT\_UDP\_REJ\_LIMIT, [56](#)
- PF\_INPUT\_x, [56](#)
- PF\_INPUT\_x\_COMMENT, [56](#)
- PF\_LOG\_LEVEL, [55](#)
- PF\_NEW\_CONFIG, [44](#)
- PF\_OUTPUT\_ACCEPT\_DEF, [58](#)
- PF\_OUTPUT\_CT\_ACCEPT\_DEF, [71](#)
- PF\_OUTPUT\_CT\_N, [71](#)
- PF\_OUTPUT\_CT\_x, [71](#)
- PF\_OUTPUT\_CT\_x\_COMMENT, [71](#)
- PF\_OUTPUT\_LOG, [58](#)
- PF\_OUTPUT\_LOG\_LIMIT, [58](#)
- PF\_OUTPUT\_N, [59](#)
- PF\_OUTPUT\_POLICY, [58](#)
- PF\_OUTPUT\_REJ\_LIMIT, [59](#)
- PF\_OUTPUT\_UDP\_REJ\_LIMIT, [59](#)
- PF\_OUTPUT\_x, [59](#)
- PF\_OUTPUT\_x\_COMMENT, [59](#)
- PF\_POSTROUTING\_N, [60](#)
- PF\_POSTROUTING\_x, [60](#)
- PF\_POSTROUTING\_x\_COMMENT, [60](#)
- PF\_PREROUTING\_CT\_ACCEPT\_DEF, [71](#)
- PF\_PREROUTING\_CT\_N, [71](#)
- PF\_PREROUTING\_CT\_x, [71](#)
- PF\_PREROUTING\_CT\_x\_COMMENT, [71](#)
- PF\_PREROUTING\_N, [61](#)
- PF\_PREROUTING\_x, [61](#)
- PF\_PREROUTING\_x\_COMMENT, [61](#)
- PF\_USR\_CHAIN\_N, [59](#)
- PF\_USR\_CHAIN\_x\_NAME, [59](#)
- PF\_USR\_CHAIN\_x\_RULE\_N, [59](#)
- PF\_USR\_CHAIN\_x\_RULE\_x, [59](#)
- PF\_USR\_CHAIN\_x\_RULE\_x\_COMMENT, [59](#)
- POWERMANAGEMENT, [28](#)
- pptp, [71](#)
- PXESUBDIR, [115](#)
- REMOTEHOSTNAME, [114](#)
- REMOTEPATHNAME, [114](#)
- REMOTEPORT, [115](#)
- REMOTEREMOUNT, [115](#)
- REMOTEUPDATE, [114](#)
- REMOTEUSERNAME, [114](#)
- RTC\_SYNC, [26](#)
- sane, [71](#)
- SER\_CONSOLE, [30](#)
- SER\_CONSOLE\_IF, [31](#)
- SER\_CONSOLE\_RATE, [31](#)
- sip, [71](#)
- snmp, [71](#)
- SQUEEZE\_SCRIPTS, [115](#)
- SSHKEYFILE, [115](#)
- SYSLOGD\_DEST\_N, [100](#)
- SYSLOGD\_DEST\_x, [100](#)
- SYSLOGD\_RECEIVER, [100](#)
- SYSLOGD\_ROTATE, [101](#)
- SYSLOGD\_ROTATE\_AT\_SHUTDOWN, [102](#)
- SYSLOGD\_ROTATE\_DIR, [102](#)
- SYSLOGD\_ROTATE\_MAX, [102](#)
- tftp, [71](#)
- TFTPBOOTIMAGE, [115](#)
- TFTPBOOTPATH, [115](#)
- TIME\_INFO, [25](#)
- VERBOSE, [114](#)
- Y2K\_DAYS, [103](#)