

**Package ISDN**  
**Version 4.0.0-testing-x86\_64-r60780**

Frank Meyer  
email: [frank@fli41.de](mailto:frank@fli41.de)

The fli41-Team  
email: [team@fli41.de](mailto:team@fli41.de)

October 5, 2022

# Contents

<b>1. Documentation For Package ISDN</b>	<b>3</b>
1.1. ISDN - Communication Over Active And Passive ISDN-Cards . . . . .	3
1.1.1. Establishing An ISDN Connection . . . . .	3
1.1.2. ISDN Card . . . . .	4
1.1.3. OPT_ISDN_COMP (EXPERIMENTAL) . . . . .	6
1.1.4. ISDN-Circuits . . . . .	7
1.1.5. OPT_TELMOND - telmond-Configuration . . . . .	15
1.1.6. OPT_RCAPID - Remote CAPI Daemon . . . . .	17
<b>A. Appendix For Package ISDN</b>	<b>19</b>
A.1. ISDN . . . . .	19
A.1.1. Technical Details About Dial-In And Routing With ISDN . . . . .	19
A.1.2. Error Messages Of The ISDN-Subsystem (i4l-Documentation) . . . . .	20
<b>List of Figures</b>	<b>23</b>
<b>List of Tables</b>	<b>24</b>
<b>Index</b>	<b>25</b>

# 1. Documentation For Package ISDN

## 1.1. ISDN - Communication Over Active And Passive ISDN-Cards

fli4l is mainly aiming to be used as an ISDN- and/or DSL-router. By setting `OPT_ISDN='yes'` the ISDN package is activated. Precondition is a ISDN-card supported by fli4l.

Default setting: `OPT_ISDN='no'`

### 1.1.1. Establishing An ISDN Connection

fli4l's behaviour during dial-in is determined by three variables `DIALMODE`, `ISDN_CIRC_X_ROUTE_X`, `ISDN_CIRC_X_TIMES`. `DIALMODE` (Page ??) (in `<config>/base.txt`) determines whether a connection will be automatically established on an active circuit on packet arrival or not. `DIALMODE` may have the following values:

**auto** If a packet reaches an ISDN-circuit (res. the ISDN interface derived from it - ipp\*) a connection will be established automatically. If and when a packet reaches an ISDN-circuit is determined by `ISDN_CIRC_X_ROUTE_X` and `ISDN_CIRC_X_TIMES`.

**manual** In manual mode the connection has to be established via `imond/imonc`. How this is done see `imonc/imond`.

**off** No ISDN connections will be established.

Which circuits packets will trigger a dial-in is defined by `ISDN_CIRC_X_ROUTE_X`. Normally this uses `'0.0.0.0/0'` as the 'default route'. This means that all packets that leave the local net are using this circuit if it is active. If and when it is active is determined by `ISDN_CIRC_X_TIMES` for fli4l is doing *least cost routing* over a predefined circuit (see *Least-Cost-Routing - Functionality* (Page ??) in the base documentation). If not all but only packets for a certain net should be routed over this circuit (i.e. a company net) additional nets can be given here. fli4l will then set a permanently active ISDN route over the interface set for this circuit. If a packet is sent to this net a connection will be automatically established.

As said before `ISDN_CIRC_X_TIMES` besides the connection costs for a circuit describes also if and when a circuit with a default route is active and can trigger a connection. 'When' is defined by the time-info, the first two elements (i.e. `Mo-Fr:09-18`), 'if' is given by the forth parameter `lc-default-route (y/n)`. fli4l (res. `imond`) will trigger a connection to the internet provider and assure that all packets leaving the local net are routed over the circuit that is active at this time.

The standard use cases in summary:

- If simply a connection to the internet is intended `DIALMODE` is set to `auto`, 1-n circuits are to be defined which have an initial route of `'0.0.0.0/0'` and whose times (times with `lc-default-route = y`) cover the whole week.

## 1. Documentation For Package ISDN

```
ISDN_CIRC_%_ROUTE_N='1'  
ISDN_CIRC_%_ROUTE_1='0.0.0.0/0'  
ISDN_CIRC_%_TIMES='Mo-Su:00-24:0.0148:Y'
```

- If besides that a connection to a company net should be used another (or more) circuits have to be defined. The route should differ from '0.0.0.0/0' to accomplish a permanently active route.

```
ISDN_CIRC_%_ROUTE_N='1'  
ISDN_CIRC_%_ROUTE_1='network/netmaskbits'  
ISDN_CIRC_%_TIMES='Mo-Su:00-24:0.0148:Y'
```

### 1.1.2. ISDN Card

The fli4l router generally supports the use of multiple ISDN cards simultaneously. However, this requires all ISDN cards to be driven by the *same driver type*. The driver type can be derived from the group in the table below the driver in question is part of. Consequently, it is e. g. no problem to use multiple mISDN-driven adapters or multiple HiSax-driven adapters (as long as sufficient resources are available), but it is not possible to use an mISDN-driven card and a HiSax-driven card at the same time.

#### ISDN\_%\_TYPE ISDN\_%\_IO ISDN\_%\_IO0 ISDN\_%\_IO1 ISDN\_%\_MEM ISDN\_%\_IRQ

Some technical data about the ISDN card is specified here.

The values in the example work for a TELES 16.3 set to IO-address 0xd80 via Dip-switches. For other settings of the switches the values have to be changed.

#### Common error (example):

```
ISDN_1_IO='240' -- right value would be: ISDN_1_IO='0x240'
```

Using IRQ 12 eventually a PS/2 mouse has to be deactivated in BIOS. Better choose another IRQ! „Good ones“ are mostly 5, 10 and 11.

ISDN\_%\_TYPE in principle follows the type numbers for HiSax drivers. Exception: non-HiSax-cards like i.e. the AVM-B1. For those the type numbers were extended (see below).

The list of possible HiSax-types is based on

linux-2.x.y/Documentation/isdn/README.HiSax.

Typ	Karte	Needed parameters
Dummy Type-Number:		
0	no driver (dummy)	none
Type numbers for remote CAPI drivers:		
160	AVM Fritz!Box Remote CAPI	ip,port
161	Melware Remote CAPI (rcapi)	ip,port
Type numbers for mISDN-drivers:		
301	HFC-4S/8S/E1 multiport cards	no parameter
302	HFC-PCI based cards	no parameter
303	HFCS-USB Adapters	no parameter
304	AVM Fritz!Card PCI (v1 and v2) cards	no parameter

## 1. Documentation For Package ISDN

Typ	Karte	Needed parameters
305	cards based on Infineon (former Siemens) chips: - Dialogic Diva 2.0 - Dialogic Diva 2.0U - Dialogic Diva 2.01 - Dialogic Diva 2.02 - Sedlbauer Speedwin - HST Saphir3 - Develo (former ELSA) Microlink PCI (Quickstep 1000) - Develo (former ELSA) Quickstep 3000 - Berkom Scitel BRIX Quadro - Dr.Neuhaus (Sagem) Niccy	no parameter
306	NetJet TJ 300 and TJ320 cards	no parameter
307	Sedlbauer Speedfax+ cards	no parameter
308	Winbond 6692 based cards	no parameter

My card is a Teles 16.3 NON-PNP ISA, this is Type=3.

For a ICN-2B-card IO and MEM have to be set, for example `ISDN_1_IO='0x320'`, `ISDN_1_MEM='0xd0000'`.

For newer Teles-PCI-card type=20 (instead of 21) has to be used. Those are shown by `cat /proc/pci` as "tiger" or similar.

For ISDN type 303 it is necessary to activate USB support. See USB - Support for USB-devices (Page ??).

If you really don't know what card is in your PC you can get tips for type numbers also from the i4l-FAQ or mailing list.

Card types that are signed „from isapnp setup“ have to be initialized by the PnP tool `isapnp` - if they really are PnP cards. See `OPT_PNP` - Installation of `isapnp` tools (Page ??).

ISDN type 0 is used if the ISDN package should be installed without an ISDN card, for example to use `imond` in a network router.

**ISDN\_%\_IP ISDN\_%\_PORT** For ISDN types 160 and 161 the variables (`ISDN_%_IP`) and (`ISDN_%_PORT`) set IP address resp. port of the device offered by a remote CAPI interface. The IP address is mandatory while the port number may be omitted: Depending on the chosen type a standard port will be set (Type 160: 5031, Type 161: 2662).

Example:

```
ISDN_1_TYPE='160' # AVM Fritz!Box
ISDN_1_IP='192.168.177.1'
```

**ISDN\_DEBUG\_LEVEL** Sets the debug level for the HiSaX driver. Debug level is concatenated by addition of the following values (cited from the original docs):

Number	Debug-Information
1	Link-level <-> hardware-level communication
2	Top state machine
4	D-Channel Q.931 (call control messages)
8	D-Channel Q.921
16	B-Channel X.75
32	D-Channel l2
64	B-Channel l2
128	D-Channel link state debugging
256	B-Channel link state debugging
512	TEI debug
1024	LOCK debug in callc.c
2048	More debug in callc.c (not for normal use)

The default setting (`ISDN_DEBUG_LEVEL='31'`) should be enough for most purposes.

**ISDN\_VERBOSE\_LEVEL** Sets the “verbosity” of the ISDN subsystem in fli4l kernel. Each verbose-level includes levels with lower numbers. Verbose levels are:

- '0' no additional informations
- '1' events triggering an ISDN connection will be logged
- '2' and '3' Calls are logged
- '4' and more Data transfer rates will be logged

Messages are sent over the kernel logging interface activated by `OPT_SYSLOGD` (Page ??).

**Important:** *If calls should be logged with telmond don't set this value lower than 2 otherwise telmond would lack informations for logging.*

Default setting: `ISDN_VERBOSE_LEVEL='2'`

**ISDN\_FILTER** Activates filtering mechanism of the kernel to achieve hangup after the specified hangup timeout. See <http://www.fli4l.de/hilfe/howtos/basteleien/hangup-problem-loesen/> for additional informations.

**ISDN\_FILTER\_EXPR** Specifies the filter to use if `ISDN_FILTER` is set to 'yes'.

### 1.1.3. OPT\_ISDN\_COMP (EXPERIMENTAL)

`OPT_ISDN_COMP='yes'` activates LZS- and BSD-compression. Credits for this go to Arwin Vos-selman (email: [arwin\(at\)xs4all\(dot\)nl](mailto:arwin(at)xs4all(dot)nl)). This addon package is in experimental state.

Default setting: `OPT_ISDN_COMP='no'`

The needed parameters for LZS-compression in detail:

**ISDN\_LZS\_DEBUG (EXPERIMENTAL)** Debug-level-settings:

- '0' no debugging informations
- '1' normal debugging informations
- '2' enhanced debugging informations
- '3' extreme debugging informations (incl. dumping of data packets)

Default setting: `ISDN_LZS_DEBUG='1'`

**ISDN\_LZS\_COMP (EXPERIMENTAL)** Compression level (not decompression!). Please use value 8. Values from 0 to 9 are possible.

Higher numbers will compress better at the cost of higher CPU load with 9 being disproportional excessive.

Default setting: `ISDN_LZS_COMP='8'`

**ISDN\_LZS\_TWEAK (EXPERIMENTAL)** Keep this value at '7' at the moment.

Default setting: `ISDN_LZS_TWEAK='7'`

Beside this three values the variable `ISDN_CIRC_x_FRAMECOMP` has to be set (see next chapter).

#### 1.1.4. ISDN-Circuits

More connections over ISDN can be defined in `fli4l` configuration. A maximum of two at a time is possible over one ISDN card.

Definition of connections is done by so-called circuits. One circuit is used per connection. In the `config.txt` example two circuits are defined:

- Circuit 1: Dialout over Internet-by-call provider Microsoft Network, Sync-PPP
- Circuit 2: Dialin/Dialout to an ISDN-router (maybe another `fli4l`) over Raw-IP, i.e. as a connection to a company net somewhere.

If `fli4l` is simply used as an internet gateway only one circuit is needed. Exception: `fli4l`'s least-cost features should be used. In this case define different circuits for all allowed timespans, see below.

**ISDN\_CIRC\_N** Sets the number of used ISDN circuits. If `fli4l` is used only to monitor incoming ISDN calls set:

```
ISDN_CIRC_N='0'
```

If `fli4l` is simply used as an internet gateway one circuit is enough. Exception: LC-routing, see below.

**ISDN\_CIRC\_x\_NAME** Set a name for the circuit - maximum length is 15 characters. The `imon` client `imonc.exe` will show this instead of the telephone number dialed. Possible characters are 'A' to 'Z' (Capitals are possible), number '0' to '9' and hyphens '-'. Example:

```
ISDN_CIRC_x_NAME='msn'
```

This name can be used in the packet filter or with OpenVPN. If for example the packet filter should control an ISDN circuit a 'circuit\_' has to prefix the circuit name. If an ISDN circuit is called 'willi' the packet filter has to be set like this:

```
PF_INPUT_3='if:circuit_willi:any prot:udp 192.168.200.226 192.168.200.254:53 ACCEPT'
```

**ISDN\_CIRC\_x\_USEPEERDNS** This determines whether the name servers transferred by the internet provider during dial-in should be filled in the configuration file of the local name server for the duration of the connection. This only makes sense for circuits used for connecting to an internet provider. Nearly all providers support this name server transfer.

After name server IP addresses have been transferred name servers entries from base.txt's *DNS\_FORWARDERS* are removed from the configuration file of the local name server and the transferred ones are filled in as forwarders. After this the name server is forced to reload its configuration. The name server cache will be preserved and names already resolved are kept.

This option has the advantage to work with the nearest possible name servers at any time, as far as the provider transmits correct IP addresses - name resolution is faster then.

In case of failing DNS servers at the provider side transmitted DNS server addresses usually are corrected rapidly by the provider.

After all it is absolutely necessary for the first dial-in to provide a valid name server in base.txt's *DNS\_FORWARDERS*. Otherwise the first request can not be resolved correctly. In addition the initial configuration of the name server will be restored on hangup.

Default setting: `ISDN_CIRC_x_USEPEERDNS='yes'`

**ISDN\_CIRC\_x\_TYPE** `ISDN_CIRC_x_TYPE` specifies the type of connection x. Possible values are:

'raw' RAW-IP  
'ppp' Sync-PPP

In most cases PPP is used, Raw-IP is a little more efficient because of the missing PPP overhead. Authentication is not possible with Raw-IP but with variable `ISDN_CIRC_x_DIALIN` (see below) limitations to explicit ISDN numbers ("Clip") can be accomplished. If `ISDN_CIRC_x_TYPE` is set to 'raw' /etc/ppp a raw up/down script will be executed in analog to the PPP up/down scripts.

**ISDN\_CIRC\_x\_BUNDLING** For ISDN channel bundeling the MPPP protocol according to RFC 1717 is used. This inherits the following mostly irrelevant limitations:

- Only possible with PPP connections, not with raw circuits
- channel bundeling according to newer RFC 1990 (MLP) is not possible

The second channel either can be added manually using imonc client or automatically by bandwidth adaption, see description for `ISDN_CIRC_x_BANDWIDTH`.

Default setting: `ISDN_CIRC_x_BUNDLING='no'`

Caution: using channel bundeling together with compression can trigger some problems, see description for `ISDN_CIRC_x_FRAMECOMP`.

**ISDN\_CIRC\_x\_BANDWIDTH** If ISDN channel bundeling is activated by `ISDN_CIRC_x_BUNDLING='yes'` an automatical addition of the second ISDN channel can be configured here. Two parameters have to be set:

1. threshold level in bytes/second (S)

2. time interval in seconds (*Z*)

If threshold level *S* is exceeded for *Z* seconds imond will add a second channel automatically. If threshold level *S* is underrun for *Z* seconds imond will deactivate the second channel again. Automatic bandwidth adaption may be deactivated with `ISDN_CIRC_1_-BANDWIDTH=""`. After that channel bundeling can only be accomplished manually by the imonc client.

Examples:

- `ISDN_CIRC_1_BANDWIDTH='6144 30'`

If the transfer rate exceeds 6 kibibyte/second for 30 seconds the second channel will be added.

- `ISDN_CIRC_1_BANDWIDTH='0 0'`

The second ISDN channel will be added immediately (not later than 10 seconds after connection establishment and stays active until the connection terminates completely.

- `ISDN_CIRC_1_BANDWIDTH=""`

The second ISDN channel only can be added manually, furthermore `ISDN_CIRC_1_-BUNDLING='yes'` has to be set.

- `ISDN_CIRC_1_BANDWIDTH='10000 30'`

This is intended to add a second channel after 30 seconds if 10000 B/s were reached during that timespan. This won't work because ISDN has a maximum tranfer rate of 8 kB/s.

If `ISDN_CIRC_x_BUNDLING='no'` is set the value in variable `ISDN_CIRC_x_BANDWIDTH` is irrelevant.

Default setting: `ISDN_CIRC_x_BANDWIDTH=""`

**ISDN\_CIRC\_x\_LOCAL** This variable holds the local IP address on the ISDN side.

This value should be **empty** if using dynamical address assignment. The IP address will be negotiated during connection establishment. In most cases internet providers hand out dynamic addresses. If a fixed IP address is used specify it here. This variable is optional and has to be added to the config file.

**ISDN\_CIRC\_x\_REMOTE** This variable holds the remote IP address and netmask on the ISDN side. Classes Inter-Domain routing (CIDR) notation has to be used. Details for CIDR (Page ??) can be found in the base documentation for `IP_NET_x`.

With dynamic address negotiation this should **empty**. The IP address will be negotiated on connection establishment. In most cases internet providers hand out dynamic addresses. If a fixed IP address is used specify it here. This variable is optional and has to be added to the config file.

The netmask provided will be used for interface configuration after dial-in. A route to the dial-in host itself will be generated as well. As you most probably won't need this route it is best to generate a direct route to the dial-in host by setting the netmask to /32. For details see [Chapter: Technical Details For Dialin](#) (Page 19).

**ISDN\_CIRC\_x\_MTU ISDN\_CIRC\_x\_MRU** With this optional variable the so-called **MTU** (maximum transmission unit) and **MRU** (maximum receive unit) can be set. Optional means that the variable has to be added manually to the configuration file by the user! Usually MTU is 1500 and MRU 1524. This settings should only be changed in rare special cases!

**ISDN\_CIRC\_x\_CLAMP\_MSS** Set this to 'yes' when using synchronous ppp (ISDN\_CIRC\_x\_TYPE='ppp') and one of the following symptoms occurs:

- Webbrowser connects to the webserver without error messages but no pages are displayed and nothing happens,
- sending of small E-mails is working but bigger ones trigger problems or
- ssh works but scp hangs after initial connection.

Default setting: ISDN\_CIRC\_x\_CLAMP\_MSS='no'

**ISDN\_CIRC\_x\_HEADERCOMP** ISDN\_CIRC\_x\_HEADERCOMP='yes' activates Van-Jacobson compression or header compression. Not all providers are supporting this. If activated compression leads to problems while dialing in set this to 'no'.

Default setting: ISDN\_CIRC\_x\_HEADERCOMP='yes'

**ISDN\_CIRC\_x\_FRAMECOMP (EXPERIMENTAL)** This parameter is only used if OPT\_ISDN\_COMP is set to 'yes'. It handles frame compression.

The following values are possible:

'no'	no frame compression
'default'	LZS according to RFC1974(std) and BSDCOMP 12
'all'	Negotiate lzs and bsdcomp
'lzs'	Negotiate lzs only
'lzsstd'	LZS according to RFC1974 Standard Mode ("Sequential Mode")
'lzsext'	LZS according to RFC1974 Extended Mode
'bsdcomp'	Negotiate bsdcomp only
'lzsstd-mh'	LZS Multihistory according to RFC1974 Standard Mode ("Sequential Mode")

You have to find out by yourself which value is supported by the provider. T-Online supports only 'lzsext' as far as I know. With most other providers 'default' should work.

Attention: using channel bundeling together with 'lzsext' can lead to problems specific to the dial-in server and provider. As providers use different types of dial-in servers there can be differences between dial-in servers of the same provider.

'lzsstd-mh' is meant for router-to-router usage (r2r). It is not used by providers but using it between two fli4l routers leads to significant improvements while transferring more files in parallel. Header compression is needed here and therefore will be activated automatically.

**ISDN\_CIRC\_x\_REMOTENAME** This variable normally is only relevant when configuring fli4l as a dial-in router. Set the name of a remote hosts if you want but this is not needed.

Default setting: ISDN\_CIRC\_x\_REMOTENAME=""

**ISDN\_CIRC\_x\_PASS** Enter provider data here. In the example data for Microsoft Network is used.

ISDN\_CIRC\_x\_USER holds the user-id, ISDN\_CIRC\_x\_PASS the password.

Note for T-Online:

Username AAAAAAAAAAATTTTTT#MMMM is composed from a 12 digit 'Anschlußkennung' plus T-Online-Number and 'Mitbenutzernummer'. Put a '#' after the T-Online-Number if it is shorter than 12 characters.

In rare cases another '#' character has to be inserted between 'Anschlußkennung' and T-Online-Number.

For T-Online-Numbers with 12 characters no additional '#' is needed.

Example: ISDN\_CIRC\_1\_USER='123456#123'

For Raw-IP circuits this variable has no meaning.

**ISDN\_CIRC\_x\_ROUTE\_N** Number of routes of this ISDN circuit. If the circuit defines a default route you must set this to '1'.

**ISDN\_CIRC\_x\_ROUTE\_X** Route(s) for this circuit. For Internet access the first entry should be '0.0.0.0/0' (default route). Format is always 'network/netmaskbits'. A host route for example would look like this: '192.168.199.1/32'. If dialing in to company or university routers name only the net you want to reach there. Examples:

```
ISDN_CIRC_%_ROUTE_N='2'  
ISDN_CIRC_%_ROUTE_1='192.168.8.0/24'  
ISDN_CIRC_%_ROUTE_2='192.168.9.0/24'
```

All nets must have an explicit entry hence for each route a new ISDN\_CIRC\_x\_ROUTE\_y=" line has to be provided.

For using fli4l's LC routing features a default route can be assigned to \*several\* circuits. Which circuit is used is driven by ISDN\_CIRC\_x\_TIMES, see below.

**ISDN\_CIRC\_x\_DIALOUT** ISDN\_CIRC\_x\_DIALOUT specifies the telephone number to be dialed. It is possible to put in several numbers (if one is busy the next is chosen) - numbers have to be separated by blanks. A maximum of five numbers can be used.

**ISDN\_CIRC\_x\_DIALIN** If the circuit (also) serves for dial-in ISDN\_CIRC\_x\_DIALIN keeps the phone number of the callee - with a region prefix but \*without\* a leading 0. Ports behind telephone systems may have to specify one or even two zeros.

If more users should be able to dial in over a circuit more numbers may be added separated by blanks. It is advised to assign a separate circuit for each caller although. Otherwise two callers trying to dial in at the same time (which is absolutetely feasible with 2 ISDN channels) could collide on behalf of IP addresses assigned.

If callers don't transfer a number during calling '0' could be used. Caution: everyone not transferring a number is allowed to call in then!

If number-independent dial-in should be realized set '\*'.

In both cases a separate authentication (see ISDN\_CIRC\_x\_AUTH) is unavoidable.

**ISDN\_CIRC\_x\_CALLBACK** Settings for callback, possible values:

'in'        fli4l is called and calls back  
'out'       fli4l calls, hangs up and waits for callback  
'off'       no callback  
'cbcp'      callback control protocol  
'cbcp0'     callback control protocol 0  
'cbcp3'     callback control protocol 3  
'cbcp6'     callback control protocol 6

CallBack control protocol (aka 'Microsoft CallBack') cbcp6 is the protocol mostly used.

Default setting: 'off'

**ISDN\_CIRC\_x\_CBNUMBER** Set a callback number for protocol cbcp, cbcp3 and cbcp6 here (mandatory for cbcp3).

**ISDN\_CIRC\_x\_CBDELAY** This variable sets a delay in seconds to be waited until triggering callback. The meaning differs depending on the direction of callback:

- ISDN\_CIRC\_x\_CALLBACK='in':

If fli4l is called and should call back ISDN\_CIRC\_x\_CBDELAY specifies the waiting time until calling back. Use ISDN\_CIRC\_x\_CBDELAY='3' as a rule of thumb. A lower value may work and speed up connection establishment then depending on whom to call back.

- ISDN\_CIRC\_x\_CALLBACK='out':

In this case ISDN\_CIRC\_x\_CBDELAY is the ringing timespan for the other party until fli4l waits for callback. ISDN\_CIRC\_x\_CBDELAY='3' is a good rule of thumb here either. On long distance calls it takes up to 3 seconds until the other router is even recognizing the call. If in doubt simply try.

If setting ISDN\_CIRC\_x\_CALLBACK='off', ISDN\_CIRC\_x\_CBDELAY is ignored. This variable is ignored with CallBack Control Protocol as well.

**ISDN\_CIRC\_x\_EAZ** In the example the MSN (called EAZ here) is set to 81330. Set your own MSN \*without\* area code here.

Depending on your telephony system only the extension station number could be necessary. Setting an additional '0' may also help sometimes.

This variable may be empty which can help with some telephony systems as well.

**ISDN\_CIRC\_x\_SLAVE\_EAZ** If fli4l is connected on the internal S0-Bus of a telephony system and you want to use channel bundeling you may have to specify a second extension station number for the slave channel.

Normally this variable can stay empty.

**ISDN\_CIRC\_x\_DEBUG** If ippdd should display additional debug informations set ISDN\_CIRC\_x\_DEBUG to 'yes'. Ippdd will log these informations to syslog then.

IMPORTANT: To use syslogd for logging OPT\_SYSLOGD has to be set to 'yes'.

(See OPT\_SYSLOGD - Program For Protocolling System Messages (Page ??)) Some

## 1. Documentation For Package ISDN

messages are logged to klogd so `OPT_KLOGD` (See `OPT_KLOGD` - Kernel-Message-Logger (Page ??)) should be set to 'yes' as well for debugging ISDN.

For Raw-IP-Circuits `ISDN_CIRC_x_DEBUG` has no meaning.

**ISDN\_CIRC\_x\_AUTH** If the circuit is also used for dial-in and an authentication over PAP or CHAP should be used by the calling party set `ISDN_CIRC_x_AUTH` to 'pap' or 'chap' - but \*only\* then. In all other cases this variable has to be empty!

Cause: An Internet provider will never authenticate with you - but there may be exceptions to this rule.

Use the entries `ISDN_CIRC_x_USER` and `ISDN_CIRC_x_PASS` for username and password.

For Raw-IP-Circuits this variable has no meaning.

**ISDN\_CIRC\_x\_HUP\_TIMEOUT** `ISDN_CIRC_x_HUP_TIMEOUT` sets the time after that `fli4l` disconnects from the provider if no traffic is detected over the connection. In the example the connection will be hung up after 40 seconds idle time to save money. On new accesses `fli4l` connects again in a short timespan. This makes sense especially with providers that have seconds charge intervals!

At least while testing you should keep an eye on `fli4l`'s automatic dial-in and hangup using either console or `imon-client`. In case of faulty configuration the ISDN connection could become an unwanted permanent line.

Setting this to '0' means that `fli4l` doesn't use idle time and won't hang up by itself anymore. Use with care!

**ISDN\_CIRC\_x\_CHARGEINT** Set charge interval in seconds which will be used for calculating online costs.

Most providers charge by minute intervals. In this case use the value '60'. For providers that charge in seconds set `ISDN_CIRC_x_CHARGEINT` to '1'.

Addition for `ISDN_CIRC_x_CHARGEINT`  $\geq$  60 seconds:

If no traffic was detected for `ISDN_CIRC_x_HUP_TIMEOUT` seconds the connection will be terminated approx. 2 seconds before reaching the `chargint` timespan. Charged time is used nearly complete this way. A really neat feature of `fli4l`.

With charging intervals of under a minute this does not make sense so this feature is only used for charging intervals of more than 60 seconds.

**ISDN\_CIRC\_x\_TIMES** Specify here when and at what charge the circuit should be active. This makes it possible to use different circuits as default routes at different daytimes (least-cost-routing). The `imond` daemon controls route-allocation.

Composition of the variable:

```
ISDN_CIRC_x_TIMES='times-1-info [times-2-info] ...'
```

Each `times-?-info` field consists of 4 subfields separated by colons (':').

### 1. Field: W1-W2

Weekday-timespan, for example Mo-Fr or Sa-Su. English and german notation are possible. If a single weekday should be specified write W1-W1, for example Su-Su.

## 1. Documentation For Package ISDN

### 2. Field: hh-hh

Daytime-timespan, for example 09-18 or also 18-09. 18-09 is equal to 18-24 plus 00-09. 00-24 means the whole day.

### 3. Field: Charge

Costs per minute in currency units, for example 0.032 for 3.2 Cent per minute. The real costs are calculated in consideration of charging intervals and displayed in imon-client then.

### 4. Field: LC-default-route

May be Y or N. Meaning:

- Y: The timespan specified will be used as default route for LC-routing. Important: in this case `ISDN_CIRC_x_ROUTE='0.0.0.0/0'` must be set in addition!
- N: The timespan specified only serves for calculating online costs but won't be used for LC-Routing.

Example:

```
ISDN_CIRC_1_TIMES='Mo-Fr:09-18:0.049:N Mo-Fr:18-09:0.044:Y Sa-Su:00-24:0.044:Y'  
ISDN_CIRC_2_TIMES='Mo-Fr:09-18:0.019:Y Mo-Fr:18-09:0.044:N Sa-Su:00-24:0.044:N'
```

Read as follows: Circuit 1 should be used on labour days evenings/nights and on the complete weekends, Circuit 2 is used on labour days from 9 AM to 6 PM.

**Important:** *timespans specified in `ISDN_CIRC_x_TIMES` have to cover the whole week. Without that no valid configuration can be generated.*

*If timespans of all LC-default-route circuits ("Y") don't cover the complete week no default route exists during the missing times. Therefore no internet connections are possible!*

Example:

```
ISDN_CIRC_1_TIMES='Sa-Su:00-24:0.044:Y Mo-Fr:09-18:0.049:N Mo-Fr:18-09:0.044:N'  
ISDN_CIRC_2_TIMES='Sa-Su:00-24:0.044:N Mo-Fr:09-18:0.019:Y Mo-Fr:18-09:0.044:N'
```

Here for labour days from 18-09 "N" is set. At this times no route to the internet exists - surfing forbidden!

Another simple example:

```
ISDN_CIRC_1_TIMES='Mo-Su:00-24:0.0:Y'
```

for those using a flatrate.

A last note concerning LC-Routing:

Bank holidays are treated as sundays.

### 1.1.5. OPT\_TELMOND - telmond-Configuration

OPT\_TELMOND determines whether the telmond server is activated or not. It listens to incoming telephone calls and signals on TCP port 5001 the caller id transmitted and called. This information can be queried and displayed by i.e. windows- or Unix/Linux imon-client (see chapter “Client-/Server-interface imond”).

An installed ISDN card is mandatory as well as a valid configuration of OPT\_ISDN.

Testing the correct function of telmond is possible with:

```
telnet fli4l 5001
```

This should display the last call and immediately close the telnet connection.

Port 5001 is only reachable from LAN. Access from outside is blocked by the firewall per default. Further access limitations are configurable via telmond variables, see below.

Default setting: `START_TELMOND='yes'`

**TELMOND\_PORT** TCP/IP-Port on which telmond listens for connections. The default setting '5001' should only be changed in rare exceptions.

**TELMOND\_LOG** `TELMOND_LOG='yes'` activates saving of all incoming calls in a file called `/var/log/telmond.log`. The content of this file can be queried with imond-Client imonc under Unix/Linux and Windows.

Different paths or logfiles splitted by clients may be configured below.

Default setting: `TELMOND_LOG='no'`

**TELMOND\_LOGDIR** If prototyping is active `TELMOND_LOGDIR` can set a different path instead of `/var/log`, for example `/boot`. The file `telmond.log` will be saved on the boot media (which has to be mounted Read/Write “rw”) then. If 'auto' is set the logfile is created in `/boot/persistent/isdn` or at another path specified by `FLI4L_UUID`. If `/boot` is not mounted Read/Write the file is created in `/var/run`.

**TELMOND\_MSN\_N** If certain calls should only be visible on some client PC's imonc a filter can be set to achieve that MSNs are only protocolled for those PCs.

If this is necessary (for example with flat sharing) the variable `TELMOND_MSN_N` holds the number of MSN filters.

Default setting: `TELMOND_MSN_N='0'`

**TELMOND\_MSN\_x** For each MSN filter a list of IP addresses has to be set which should be able to view the call informations.

The variable `TELMOND_MSN_N` determines the number of those configurations, see above.

Composition of the variable:

```
TELMOND_MSN_x='MSN IP-ADDR-1 IP-ADDR-2 ...'
```

A simple example:

```
TELMOND_MSN_1='123456789 192.168.6.2'
```

## 1. Documentation For Package ISDN

If a call for a certain MSN should be displayed on more computers their IP addresses have to be specified one after the other.

```
TELMOND_MSN_1='123456789 192.168.6.2 192.168.6.3'
```

**TELMOND\_CMD\_N** If a telephone call (Voice) is coming in for a MSN some commands can be executed on the fli4l router optionally. **TELMOND\_CMD\_N** holds the number of commands to be executed.

**TELMOND\_CMD\_x** **TELMOND\_CMD\_1** bis **TELMOND\_CMD\_n** holds commands to be executed for an incoming phone call.

Variable **TELMOND\_CMD\_N** specifies the quantity of commands, see above.

Composition of the variable:

```
MSN CALLER-NUMBER  COMMAND ...
```

Set the MSN without area prefix. **CALLER-NUMBER** takes the complete caller id with area prefix. If **CALLER-NUMBER** is set to an asterisk (\*) telmond won't pay attention to the caller id.

An example:

```
TELMOND_CMD_1='1234567 0987654321 sleep 5; imonc dial'  
TELMOND_CMD_2='1234568 * switch-on-coffee-machine'
```

In the first case the command sequence “sleep 5; imonc dial” is executed if caller with id 0987654321 calls MSN 1234567. Two commands are executed. At first fli4l will wait for 5 seconds for the ISDN channel to become available. After that the fli4l client imonc is started with the argument “dial”. imonc passes this command to the telmond server which will establish a network connection on the default route circuit. Which commands the imonc client can pass to the imond server is described in chapter “Client-/Server interface imond”. **OPT\_IMONC** from the package “tools” has to be installed to get this working.

The second command “switch-on-coffee-machine” will be executed if a call on MSN 1234568 comes in independent on caller id. The command “switch-on-coffee-machine” does not exist for fli4l (at the moment)!

On execution of command the following placeholders may be used:

%d	date	Date
%t	time	Time
%p	phone	Caller ID
%m	msn	Own MSN
%%	percent	the percent sign

This data can be used by the programs called, for example for sending E-Mail.

**TELMOND\_CAPI\_CTRL\_N** If using a CAPI capable ISDN adapter or a remote CAPI (type 160 or 161) it may be necessary to configure the CAPI controller on which telmond is listening for calls more explicitly. For example the Fritz!Box offers access to up to five different controllers which may not even differ (see informations at [http://www.wehavemorefun.de/fritzbox/CAPI-over-TCP#Virtuelle\\_Controller](http://www.wehavemorefun.de/fritzbox/CAPI-over-TCP#Virtuelle_Controller)). To limit the number of controllers to be used you may set the quantity. In the following array-variables TELMOND\_CAPI\_CTRL\_% it may be set which controllers are to be used.

If you don't use this variable telmond will listen on *all* available CAPI controllers.

**TELMOND\_CAPI\_CTRL\_x** If TELMOND\_CAPI\_CTRL\_N is unequal to zero the indices for the CAPI controllers have to be specified on which telmond should monitor incoming calls.

Example for the remote CAPI of a Fritz!Box with "real" ISDN connection:

```
TELMOND_CAPI_CTRL_N='2'  
TELMOND_CAPI_CTRL_1='1' # listen to incoming ISDN calls  
TELMOND_CAPI_CTRL_2='3' # listen to calls on the internal S0-Bus
```

Example for the remote CAPI of a Fritz!Box with analog connection and SIP-Forwarding:

```
TELMOND_CAPI_CTRL_N='2'  
TELMOND_CAPI_CTRL_1='4' # listen to incoming analog calls  
TELMOND_CAPI_CTRL_2='5' # listen to incoming SIP-calls
```

### 1.1.6. OPT\_RCAPID - Remote CAPI Daemon

This OPT configures the program rcapid on the fli4l router which offers access to the ISDN CAPI interface of the routers via network. Appropriate tools can access the ISDN card of the routers via network as if it was installed locally. This is similar to the package "mtgcapri". The difference is that only Windows systems can use "mtgcapri" as a client while the network interface of rcapid is only supporting linux systems at the time of writing. So both packages are ideal complements in mixed Windows- and Linux environments.

#### Konfiguration des Routers

**OPT\_RCAPID** This variable activates offering of the router's ISDN-CAPI for remote clients. Possible values are "yes" and "no". If set to "yes" the internet daemon inetd will start the rcapid daemon on incoming queries on rcapid port 6000 (port may be changed using variable RCAPID\_PORT).

Example: OPT\_RCAPID='yes'

**RCAPID\_PORT** This variable holds the TCP port to be used by the rcapid daemon.

Default setting: RCAPID\_PORT='6000'

## Configuration Of Linux Clients

To use the remote CAPI interface on a Linux PC the modular libcap20 library must be used. Actual Linux distributions install such a CAPI library (i.e. Debian Wheezy). If not the sources may be downloaded from [http://ftp.de.debian.org/debian/pool/main/i/isdnutils/isdnutils\\_3.25+dfsg1.orig.tar.bz2](http://ftp.de.debian.org/debian/pool/main/i/isdnutils/isdnutils_3.25+dfsg1.orig.tar.bz2). After unpacking and changing to the directory “cap20” the CAPI library may be compiled and installed with “./configure”, “make” and “sudo make install” as usual. If the library is installed the configuration file `/etc/cap20.conf` has to be adapted to specify the client on which rcapd is running. If the router for example is reached by the name of “fli4l” the conf file will look as follows:

```
REMOTE fli4l 6000
```

That’s all! If the program “capinfo” is installed on the linux client (part of cap4k-utils-package of many distributions) you can test the remote CAPI interface:

```
kristov@peacock ~ $ capinfo
Number of Controllers : 1
Controller 1:
Manufacturer: AVM Berlin
CAPI Version: 1073741824.1229996355
Manufacturer Version: 2.2-00 (808333856.1377840928)
Serial Number: 0004711
BChannels: 2
[...]
```

Under “Number of Controllers” the quantity of ISDN cards is displayed which are usable on the client. If this reads “0” the connection to the rcapd program is working but the ISDN card is not recognized on the router. If the connection to the rcapd program is not working at all (maybe `OPT_RCAPID` is set to “no”) an error message “cap not installed - Connection refused (111)” will be displayed. In this case check your configuration once more.

# A. Appendix For Package ISDN

## A.1. ISDN

### A.1.1. Technical Details About Dial-In And Routing With ISDN

This chapter is interesting for those who want to know what is happening 'under the hood', having special wishes for configuration or simply looking for solutions to their problems. All others are *not* encouraged to read this chapter.

After establishing a connection to the provider the `ippd` daemon that has made the connection newly configures the interface to set the negotiated IP addresses. The linux kernel automatically sets corresponding routes for remote IP address and netmask. Existing special routes will be deleted. If no netmask is given the `ippd` derives it from the remote IP netmask (Class A, B and C subnets will be used here). The vanishing of existing and appearing of new routes has raised problems time and time again:

- Company networks were unaccessible because of routes vanishing or being overlaid by newly set ones
- Interfaces were dialing in apparently without cause because a packet was routed by the kernel to another interface instead of the default route
- ...

Because of that it is tried to avoid unwanted routes.

The following things will be changed to achieve this:

- remote IP will be set to 0.0.0.0 if nothing else is specified. Hence the routes configured by the kernel while initializing the interface will vanish.
- additionally set routes will be saved in a file
- if a netmask is given for the circuit it will be transferred to the `ippd` in order to use it for the configuration of the interface (and therefore route generation) after negotiation of an IP.
- after dial-in the saved routes of the circuit will be reloaded from the file and set again (they were deleted by the kernel while `ippd` was reconfiguring the interface)
- after hangup the interface will be reconfigured and routes are set anew to restore the initial situation.

Configuration of a circuits looks like this in that case:

- item default route

## A. Appendix For Package ISDN

```
ISDN_CIRC_%_ROUTE='0.0.0.0'
```

If the circuit is a lcr circuit and “active” in the moment a default route will be set towards the circuit (res. the according interface). After dial-in a host-route to the provider appears that vanishes after hanging up.

- special routes

```
ISDN_CIRC_%_ROUTE='network/netmaskbits'
```

The given routes for the circuit (res. the according interface) will be set. After dial-in the routes deleted by the kernel will be restored and a host-route to the dial-in node exists. After hangup the initial state will be restored.

- remote ip

```
ISDN_CIRC_%_REMOTE='ip address/netmaskbits'  
ISDN_CIRC_%_ROUTE='network/netmaskbits'
```

While configuring the interface routes to the target net appear (according to IP address AND netmask). If the specified IP is kept after dial-in (meaning no other IP is negotiated during connection establishment) the route will be kept as well.

If another IP was negotiated during dial-in the route will change accordingly (new IP AND netmask).

For additional routes see above.

This will hopefully solve *all* problems raised by special routes. The way of correction may change in the future but the principle won't.

### A.1.2. Error Messages Of The ISDN-Subsystem (i4I-Documentation)

Following is an excerpt from the Isdn4Linux Documentation (man 7 isdn\_cause).

Cause messages are 2-byte information elements, describing the state transitions of an ISDN line. Each cause message describes its origination (location) in one byte, while the cause code is described in the other byte. Internally, when EDSS1 is used, the first byte contains the location while the second byte contains the cause code. When using 1TR6, the first byte contains the cause code while the location is coded in the second byte. In the Linux ISDN subsystem, the cause messages visible to the user are unified to avoid confusion. All user visible cause messages are displayed as hexadecimal strings. These strings always have the location coded in the first byte, regardless if using 1TR6 or EDSS1. When using EDSS1, these strings are preceded by the character 'E'.

**LOCATION** The following location codes are defined when using EDSS1:

- 00 Message generated by user.
- 01 Message generated by private network serving the local user.
- 02 Message generated by public network serving the local user.
- 03 Message generated by transit network.
- 04 Message generated by public network serving the remote user.
- 05 Message generated by private network serving the remote user.
- 07 Message generated by international network.
- 0A Message generated by network beyond inter-working point.

## A. Appendix For Package ISDN

**CAUSE** The following cause codes are defined when using EDSS1:

- 01 Unallocated (unassigned) number.
- 02 No route to specified transit network.
- 03 No route to destination.
- 06 Channel unacceptable.
- 07 Call awarded and being delivered in an established channel.
- 10 Normal call clearing.
- 11 User busy.
- 12 No user responding.
- 13 No answer from user (user alerted).
- 15 Call rejected.
- 16 Number changed.
- 1A Non-selected user clearing.
- 1B Destination out of order.
- 1C Invalid number format.
- 1D Facility rejected.
- 1E Response to status enquiry.
- 1F Normal, unspecified.
- 22 No circuit or channel available.
- 26 Network out of order.
- 29 Temporary failure.
- 2A Switching equipment congestion.
- 2B Access information discarded.
- 2C Requested circuit or channel not available.
- 2F Resources unavailable, unspecified.
- 31 Quality of service unavailable.
- 32 Requested facility not subscribed.
- 39 Bearer capability not authorised.
- 3A Bearer capability not presently available.
- 3F Service or option not available, unspecified.
- 41 Bearer capability not implemented.
- 42 Channel type not implemented.
- 45 Requested facility not implemented.
- 46 Only restricted digital information bearer.
- 4F Service or option not implemented, unspecified.
- 51 Invalid call reference value.
- 52 Identified channel does not exist.
- 53 A suspended call exists, but this call identity does not.
- 54 Call identity in use.
- 55 No call suspended.
- 56 Call having the requested call identity.
- 58 Incompatible destination.
- 5B Invalid transit network selection.
- 5F Invalid message, unspecified.
- 60 Mandatory information element is missing.
- 61 Message type non-existent or not implemented.
- 62 Message not compatible with call state or message or message type non existent or not implemented.
- 63 Information element non-existent or not implemented.
- 64 Invalid information element content.
- 65 Message not compatible.
- 66 Recovery on timer expiry.

*A. Appendix For Package ISDN*

- 6F Protocol error, unspecified.
- 7F Inter working, unspecified.

# List of Figures

# List of Tables

# Index

ISDN\_%\_IO, 4  
ISDN\_%\_IO0, 4  
ISDN\_%\_IO1, 4  
ISDN\_%\_IP, 5  
ISDN\_%\_MEM, 4  
ISDN\_%\_PORT, 5  
ISDN\_%\_TYPE, 4  
ISDN\_CIRC\_N, 7  
ISDN\_CIRC\_x\_AUTH, 13  
ISDN\_CIRC\_x\_BANDWIDTH, 8  
ISDN\_CIRC\_x\_BUNDLING, 8  
ISDN\_CIRC\_x\_CALLBACK, 11  
ISDN\_CIRC\_x\_CBDELAY, 12  
ISDN\_CIRC\_x\_CBNUMBER, 12  
ISDN\_CIRC\_x\_CHARGEINT, 13  
ISDN\_CIRC\_x\_CLAMP\_MSS, 10  
ISDN\_CIRC\_x\_DEBUG, 12  
ISDN\_CIRC\_x\_DIALIN, 11  
ISDN\_CIRC\_x\_DIALOUT, 11  
ISDN\_CIRC\_x\_EAZ, 12  
ISDN\_CIRC\_x\_FRAMECOMP, 10  
ISDN\_CIRC\_x\_HEADERCOMP, 10  
ISDN\_CIRC\_x\_HUP\_TIMEOUT, 13  
ISDN\_CIRC\_x\_LOCAL, 9  
ISDN\_CIRC\_x\_MRU, 9  
ISDN\_CIRC\_x\_MTU, 9  
ISDN\_CIRC\_x\_NAME, 7  
ISDN\_CIRC\_x\_PASS, 10  
ISDN\_CIRC\_x\_REMOTE, 9  
ISDN\_CIRC\_x\_REMOTENAME, 10  
ISDN\_CIRC\_x\_ROUTE\_N, 11  
ISDN\_CIRC\_x\_ROUTE\_X, 11  
ISDN\_CIRC\_x\_SLAVE\_EAZ, 12  
ISDN\_CIRC\_x\_TIMES, 13  
ISDN\_CIRC\_x\_TYPE, 8  
ISDN\_CIRC\_x\_USEPEERDNS, 8  
ISDN\_CIRC\_x\_USER, 10  
ISDN\_DEBUG\_LEVEL, 5  
ISDN\_FILTER, 6  
ISDN\_FILTER\_EXPR, 6  
ISDN\_LZS\_COMP, 6  
ISDN\_LZS\_DEBUG, 6  
ISDN\_LZS\_TWEAK, 7  
ISDN\_VERBOSE\_LEVEL, 6  
OPT\_ISDN, 3  
OPT\_ISDN\_COMP, 6  
OPT\_RCAPID, 17  
OPT\_TELMOND, 15  
RCAPID\_PORT, 17  
TELMOND\_CAPI\_CTRL\_N, 16  
TELMOND\_CAPI\_CTRL\_x, 17  
TELMOND\_CMD\_N, 16  
TELMOND\_CMD\_x, 16  
TELMOND\_LOG, 15  
TELMOND\_LOGDIR, 15  
TELMOND\_MSN\_N, 15  
TELMOND\_MSN\_x, 15  
TELMOND\_PORT, 15