PPP

COLLABORATORS

	TITLE : PPP		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 23, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

PPP

1.1 PPP.guide

                   PPP
***

   This is the documentation for PPP V1.30, a SANA-II compatible PPP
device. Copyright (C) 1994,1995 Holger Kruse. All rights reserved.

1.2 PPP.guide/NODE_DISCLAIMER

Disclaimer
****

PPP IS SUPPOSED TO BE A SANA-II COMPATIBLE PPP DEVICE THAT CAN BE USED
TO CONNECT YOUR AMIGA TO ANY OTHER PPP HOST OVER A SERIAL LINE. EVEN
THOUGH EVERY EFFORT HAS BEEN MADE TO MAKE PPP AS COMPATIBLE TO THE
SANA-II AND PPP STANDARDS AS POSSIBLE, I CANNOT RULE OUT THE
POSSIBILITY THAT PPP HAS BUGS THAT HAVE HARMFUL SIDE EFFECTS ON YOUR
SYSTEM OR ON THE HOST YOUR ARE CONNECTED TO.

I HEREBY REJECT ANY LIABILITY OR RESPONSIBILITY FOR THESE OR ANY OTHER
CONSEQUENCES FROM THE USE OF PPP WHATSOEVER. THIS INCLUDES, BUT IS NOT
LIMITED TO, DAMAGE TO YOUR EQUIPMENT, TO YOUR DATA, TO THE PPP HOST YOU
ARE CONNECTED TO, ANY EQUIPMENT CONNECTED TO THAT HOST, PERSONAL
INJURIES, FINANCIAL LOSS OR ANY OTHER KINDS OF SIDE EFFECTS.

PPP IS PROVIDED AS-IS. THIS MEANS I DO NOT GUARANTEE THAT PPP IS FIT
FOR ANY SPECIFIC PURPOSE AND I DO NOT GUARANTEE ANY BUG FIXES, UPDATES
OR HELP DURING ERROR RECOVERY.

1.3 PPP.guide/NODE_CONDITIONS

Usage / Copying
****

   PPP is shareware. In this case this means that there are three
different versions of the program:

   * an unregistered freely distributable (under the conditions outlined
     below) version that does not have most of the more powerful PPP
     options.

   * a version for registered users that has all implemented PPP options
     enabled and usually provides a much higher throughput than the
     unregistered version. This version requires a personalized key file
     that registered users receive from me. The key file may NOT be made
     available to other users ! Giving the key file to other users or
     using key files that you did not receive directly from me for your

personal use is considered an act of software piracy ! The device
itself (without the keyfile) is freely distributable (under the
conditions outlined below), just like the unregistered version.

* a special version for distribution with commercial Amiga networking
  software. If you are producing or distributing Amiga networking
  software and would like to obtain a license to include 'ppp.device'
  in your software, please contact me. (See
              Shareware registration
                  for my address.)

    If you want to distribute the unregistered of registered version of
PPP, the following conditions apply:

* The sales price must not be higher than the cost of an (empty)
  disk plus a nominal copying fee plus costs for shipping. The total
  price must not be higher than 6 US$ or 10 DM or the equivalent in
  any other currency.

* All parts of the program and the documentation must be complete.
  The distribution of single parts or incomplete subsets of the
  original distribution is not allowed. Distribution of keyfiles is
  not allowed.

* PPP or parts of it may not be sold in combination with or as part
  of commercial software. If you would like to distribute a PPP
  device together with your Amiga networking software, please
  contact me ! You can obtain a license from me to include the full
  PPP version with your software. This would be a special version
  licensed to your company for use with one software package only.
  It is neither identical to the evaluation version nor to the
  version for registered users. However until you have my written
  approval, do not assume that you can distribute any PPP device
  version with your software.

* Program and documentation may not be changed in any way.
  Exception (this means: acceptable) is the use of archivers such as
  LHA and packers like Imploder or Powerpacker, as long as it
  remains possible to retrieve the original program/data.

1.4 PPP.guide/NODE_REGISTRATION

Registration
****

    If you often use ppp.device to connect your Amiga to other machines,
I suggest you obtain the registered ppp.device version from me. It has
many features which are not present in the evaluation version and
usually gives you a considerably higher throughput and better
interactive response time on PPP links.

    To register, print the enclosed form REGISTRATION and fill it out,
then send it to the address on top of the form along with the

registration fee of US$ 15.

   The only acceptable methods of payment are:

   * cash in US$ or DM

   * a check or money order in US$ drawn on a US bank

   * an international postal money order in US$. This is probably the
     easiest and safest way for European customers. You can buy one at
     your local post office.

   Please do NOT send EuroCheques !

   Please do NOT make any kind of payment in a currency other than US$
or DM !

   Unless you suggest otherwise I will send you the key file and the
latest registered version on an 880kB floppy disk.

   If you have Internet access and some secure information channel
(PGP/RIPEM etc.) you can get the latest archive by ftp, and I could
send you the keyfile by encrypted e-mail. In this case please send me
your key fingerprint (for PGP) or MD5 key digest (for RIPEM) and a
pointer to your public key along with your registration.

   I assure you that the information you provide will be treated
confidentially.

   Thank you very much in advance !

   If you send me your registration during the US summer term (May –
July), please expect a LONG delay until you receive answer. I might not
be at home for one or two months – sorry !

   My address is:

     Holger Kruse
     12006 Coed Drive
     Orlando, FL 32826
     USA

     Internet: kruse@cs.ucf.edu
     BIX: hkruse

1.5 PPP.guide/NODE_INTRODUCTION

                 Introduction
****

   This archive contains a SANA-II compatible implementation of PPP for
Amiga computers.

   For your information:

SANA-II

General information about SANA-II

PPP

General information about PPP

PPP vs. SLIP

Comparison of PPP and SLIP

   This implementation of ppp.device has so far only been tested and
used with AmiTCP/IP although it probably works with AS225 and Envoy,
too. The information about installation and configuration applies to
AmiTCP/IP only, so if you want to use ppp.device with AS225 or Envoy
you have to figure these things out for yourself...

   At this time the device does not support incoming calls in any way,
i.e. it can probably not be combined with getty or similar programs in
any useful manner, and all PPP configuration options (like IP number
negotiation, authentication etc.) assume a client-like operation, i.e.
they assume that you are the one who attempts to connect your Amiga to
the Internet over the PPP link.

   This also means that you cannot use the current ppp.device to build
a mailbox-like environment on your Amiga that waits for incoming calls,
assigns IP numbers to callers and provides them with a TCP/IP
connection to your Amiga. It is however possible to use ppp.device to
connect two Amigas by a PPP link, if you have access to both Amigas,
i.e. if you can power them up and start ppp.device on them at about the
same time.

   This limitation only applies to the way a connection is established.
Once the connection is up and running, your Amiga can act as both
client and server in the usual way, i.e. it is for instance possible to
telnet into your Amiga from outside, if you have setup your protocol
stack accordingly, or to run an X-Windows server like DaggeX on your
Amiga and access it from the other side of the PPP link.

   Currently the device only supports protocol stacks that are based on
IP, such as AS225, AmiTCP/IP and Envoy, or on DECnet, such as EnlanDFS.
Other protocol stacks for which PPP specifications exist (like
AppleTalk, Novell IPX or ISO/OSI) are not supported yet. If you know of
any such protocol stack that is SANA-II compatible I would appreciate
some information about it.

   Since version 1.17 ppp.device has a builtin dialer that can be used
to dial up a connection, send userid and password to the host and
select PPP mode on the host. See
                The builtin dialer
                 for details how to
use the dialer.

   If you do not want to use the builtin dialer, you can still use
terminal programs, ARexx scripts etc. for dialing and only start
ppp.device when the connection has been established.

1.6 PPP.guide/NODE_SANAII

```
SANA-II
=======
```

   SANA-II is a standard specified by Commodore that describes the
interface between a networking protocol stack (such as AmiTCP/IP, AS225
or Envoy) and the underlying network device (such as Ethernet, Arcnet
or a serial RS-232 line running SLIP or PPP).

   This ppp.device is compatible with SANA-II Rev 2.0, which has been
published by Commodore in early 1994, but can also be used with
programs that use earlier SANA-II specifications.

   In theory any SANA-II compatible protocol stack can be combined with
any SANA-II compatible network device to transmit packets over the
corresponding hardware. Most networking software available for Amiga is
SANA-II compatible. The most notable exception is AmigaNOS (KA9Q), so
that program cannot be combined with the enclosed ppp.device.

   You can run Envoy and AS225R2 over the same link simultaneously
(because of a 'hack' in Envoy), but not Envoy and AmiTCP/IP 2.3/3.0b.
This is not a limitation in the PPP driver, but in the old SANA-II
specifications, and the same problem exists with other drivers like
SLIP, too. This PPP implementation already conforms to the new SANA-II
specs which allows multiple IP protocol stacks over the same line, so
once Envoy and AmiTCP/IP get updated to conform to the new SANA-II
standard, you WILL be able to run them over a PPP link simultaneously.
I believe that the new AmiTCP/IP version 4.1 (and perhaps the 4.0 demo,
too) already conform to this standard if you use the "FILTER" switch in
the "interfaces" file.

1.7 PPP.guide/NODE_PPP

```
PPP
===
```

   PPP (Point to Point Protocol) is an Internet draft standard for
transmission of network packets over serial lines, so it is similar in
purpose to the older SLIP protocol.

   The PPP specifications are scattered over several RFCs which are
available by anonymous FTP from several sources, e.g. from
ftp.internic.net. The RFCs used for this PPP implementation are: 1144,
1171, 1172, 1331, 1332, 1333, 1334, 1548, 1549, 1570, 1661 and 1662.

   The PPP standard specifies many optional features that programmers
can choose to implement. Most of these features are intended to improve

performance and increase throughput, particularly over slow serial
lines.

    The registered version of ppp.device supports most of these
features. The unregistered evaluation version only supports the most
basic features. They should be enough to get a connection up and
running, but the performance will most likely be poor. To get the most
out of your PPP link you should obtain the registered PPP version.
Besides some additional features the registered version is also
considerably faster than the evaluation version.

    The PPP protocol is still under development and the IETF PPP work
group constantly improves and enhances the standard. I will try to keep
up with these improvements and implement new features in the registered
version of ppp.device when they become finalized. Some of the features
I have planned for future versions of ppp.device are:

    * Compression. Currently the IETF PPP work group is looking at
      various ways to provide "real" transparent data compression to PPP
      links. I will probably implement some of them when they are
      finalized. Currently the standardization process for compression
      is being delayed by some patent issues.

    * Better support for PPP over ISDN.

1.8 PPP.guide/NODE_PPPSLIP

PPP vs. SLIP
============

    PPP and SLIP are quite similar in purpose, but very different in
implementation.

    SLIP is a 'non-standard', i.e. it has not been made an official
standard because of its shortcomings. However many software packages
still use SLIP (or CSLIP) for serial links.

    SLIP is completely incompatible to PPP. Some software packages
support PPP only, some support SLIP only and some support both.
However only one protocol is used at any time, so there is no such
thing as a combined PPP/SLIP.

    PPP has been designed to be a 'better SLIP', because SLIP lacks
several important features, particularly

    * SLIP does not have any checksums, so transmission errors must be
      detected by the protocol stack, which is not always feasible.

    * SLIP can only be used with up to two protocol stacks: one TCP/IP
      protocol stack and one DECnet protocol stack. It is not possible
      to run other protocol stacks (like AppleTalk, Novell etc.) over
      the same link at the same time.

    * SLIP is not extensible, i.e. it is not possible to improve it in

any (compatible) way.

   * SLIP does not know about IP numbers (or TCP/IP at all, for that
     matter), so it cannot help to negotiate IP numbers for dialup
     serial lines.

   * SLIP requires absolutely transparent lines, which makes it
     impossible to use SLIP over lines that require XON/XOFF software
     handshaking for instance.

   PPP provides solutions for all of these problems and is very likely
to become the official Internet standard for packet transport over
serial lines in the future.

   As far as performance is concerned: PPP is usually faster than SLIP
for TCP/IP connections like telnet and ftp, but slightly slower than
CSLIP. (However this applies only to the registered version of this
implementation. The unregistered PPP version may be slower than SLIP.)

   Slightly reduced performance is really the only price you might have
to pay for the more advanced PPP as compared to SLIP/CSLIP, and even
this may change when data compression becomes a part of PPP. Then PPP
may become even faster than CSLIP. Even now PPP offers you:

   * automatic IP number negotiation and thus easier configuration

   * checksums to detect transmission errors

   * DECnet protocol in addition to TCP/IP.

1.9 PPP.guide/NODE_REQUIREMENTS

Requirements
****

   You need

   * Any Amiga running Kickstart 2.04 and Workbench 2.04 or higher.
     Kick2.04/WB2.1 and Kick3.1/WB3.1 were used for testing, but other
     configurations should work as well.

   * If you want to use ppp.device with a TCP/IP protocol stack you need
     AmiTCP/IP V2.3 or higher. The device WILL NOT WORK with AmiTCP/IP
     V2.2 or earlier versions ! AmiTCP/IP V3.0beta or higher is
     strongly recommended, because this makes the setup easier for you.
     In particular: Some configurations require the program resolve
     that is included in the AmiTCP/IP V3.0beta archive. Since
     ppp.device version 1.14 configuration programs for both AmiTCP/IP
     V2.3 and V3.0beta are included in the distribution. AmiTCP/IP 4.0
     (demo) and 4.1 (commercial) are supported, too.

   * Instead of AmiTCP/IP the device will probably work with AS225 or
     Envoy, but that has not been tested yet.

* If you want to use ppp.device with DECnet, you need the commercial
  Enlan-DFS.

* A 68020/68EC020 processor or higher is strongly recommended. The
  archive contains versions for 68000/68010 CPUs and 68020+ CPUs.

1.10 PPP.guide/NODE_INSTALLATION

                    Installation
****

    Sorry, I have not written one of those neat Installer scripts for
PPP, because the installation may be quite different for each system.

    Here is what to do:

* To install the evaluation version, choose one of the files
  devs/ppp.device.000.eval (for 68000/68010) or
  devs/ppp.device.020.eval (for 68020/68EC020 or higher) depending
  on your processor type, and copy it to devs:networks/ppp.device.

* To install the registered version, choose one of the files
  devs/ppp.device.000 (for 68000/68010) or devs/ppp.device.020 (for
  68020/68EC020 or higher) (which are in a separate archive)
  depending on your processor type, and copy it to
  devs:networks/ppp.device. Then copy the keyfile you received from
  me (PPP.key) to s:. The key file is your personal ID with your
  name and serial number in it. Make sure that nobody has access to
  it, particularly if you allow telnet or ftp dialins to your Amiga
  from outside !

* Copy the contents of the bin/ directory to a location that is in
  your path, preferably to AmiTCP:bin/ if you use AmiTCP/IP.

* If you use the registered PPP version and your host requires PAP
  authentication, create a PAP password file now. (See

            PAP password file
).

* If you use the registered PPP version and your host requires CHAP
  authentication, create a CHAP password file now. (See

            CHAP password file
).

* If you use AmiTCP/IP 4.1 (commercial) you can skip the following
  step (i.e. you do not need to create a configuration file for
  ppp.device).

  Create a configuration file for the device. The name must be
  ENV:SANA2/ppp0.config, where 0 is the unit number for ppp.device
  (usually '0' - this is NOT the unit number of your serial.device).
  You should copy the file to ENVARC:SANA2/, too, to make the change

permanent. The overall file format is similar to the file format
used by the SLIP driver as described in the AmiTCP/IP
documentation. For a list of all options see
                Configuration
                , but a
single line
        serial.device 0 57600 0.0.0.0 7WIRE
probably gets you started. Three comments here:

    – Replace serial device 0 with the device name and unit number
      of your serial device.

    – Replace 57600 with the baud rate you want to use on your
      serial port. Note: This speed MUST be identical to the speed
      between computer and modem that you used to establish the
      connection.

    – If you use ppp.device with a TCP/IP protocol stack and you
      know your permanent IP number, replace 0.0.0.0 with that
      number. If you do not know your IP number or if it is
      assigned to you dynamically for each call, do not change
      0.0.0.0. This tells ppp.device to ask the remote host for
      your IP number each time the link is negotiated.

  * The final steps depend on the protocol stack you want to use.

                Additional installation for AmiTCP/IP 2.3

                Additional installation for AmiTCP/IP 3.0 beta

                Additional installation for AmiTCP/IP 4.0 demo

                Additional installation for AmiTCP/IP 4.1 commercial

                Additional installation for AS-225

                Additional installation for Envoy

                Additional installation for EnlanDFS
                    * To close your connection, interrupt all servers and clients, ←
                       exit
your protocol stack (for AmiTCP/IP: type "stopnet") and type
        offline ppp.device 0
which also drops the DTR line (unless other programs have opened
the device is shared mode), so the modem might hang up depending
on its configuration.

1.11 PPP.guide/NODE_INST_AMITCPTT

Additional installation for AmiTCP/IP 2.3
===

```
 * Add the line
        WITH netdb-myhost-tmp
   to your AmiTCP:db/netdb file. REMOVE any lines starting with HOST
   (like HOST 1.2.3.4 name) from the files AmiTCP:db/hosts,
   AmiTCP:db/netdb or any other file that AmiTCP:db/netdb includes.
   IP numbers for PPP are usually dynamic, so they CANNOT be
   specified in the permanent AmiTCP/IP database.

 * If you DO NOT KNOW the host name of your local Amiga (including
   the complete domain name, like myamiga.cc.uni.edu) or if it is
   assigned to you dynamically for each connection (this is usually
   the case if your PPP host uses dummy names like
   dialup26.cc.uni.edu for each dialup line): Create a script
   file AmiTCP:bin/startamitcp that contains the following commands:
        online devs:networks/ppp.device 0
        PPPConfigAmiTCP23 0
        run <nil: >nil: AmiTCP:AmiTCP
        SYS:Rexxc/WaitForPort AMITCP
        wait 2 secs
        execute T:AmiTCP.tmp

 * Else (if you KNOW the host name of your local Amiga): Assuming the
   name is myamiga.cc.uni.edu: Create a script file
   AmiTCP:bin/startamitcp that contains the following commands:
        online devs:networks/ppp.device 0
        PPPConfigAmiTCP23 0 myamiga.cc.uni.edu
        run <nil: >nil: AmiTCP:AmiTCP
        SYS:Rexxc/WaitForPort AMITCP
        wait 2 secs
        execute T:AmiTCP.tmp
```

A word of explanation here: If you DO NOT specify a host name with
the PPPConfigAmiTCP23 command the resolve command is used to look up
your host name from a domain name server after AmiTCP/IP has been
started. This only works if your PPP service provider has actually
assigned a name to your PPP port, and if you have set up the IP address
of your domain name server correctly in the file
AmiTCP:db/netdb-myhost. Since the "resolve" command is not included
in the AmiTCP/IP 2.3 package you have to get it from kampi.hut.fi or
another ftp site yourself.

PPPConfigAmiTCP23 creates a temporary script file T:AmiTCP.tmp that
contains the following lines to configure AmiTCP/IP for PPP:

```
   AmiTCP:bin/ifconfig lo/0 localhost
   AmiTCP:bin/ifconfig ppp.device/<num> <localip> <remoteip>
   AmiTCP:bin/route add <localip> localhost
   AmiTCP:bin/route add default <remoteip>
   rx T:AmiTCP.rexx
```

where <num>, <localip> and <remoteip> are dynamically replaced by
the correct values.

These statements configure a minimum AmiTCP/IP environment using the
PPP link as the default link. If you want to use other network devices
in addition to PPP, either add the necessary configuration statements
to the file AmiTCP:bin/startamitcp, or modify and recompile the program
PPPConfigAmiTCP23. The SAS/C source code is included in the directory

src/.

   The line rx T:AmiTCP.rexx is only added if your host name is
assigned to you dynamically. In this case PPPConfigAmiTCP23 creates a
temporary ARexx file that tells AmiTCP/IP about your host name and IP
number AFTER startup. Otherwise PPPConfigAmiTCP23 simply adds the
appropriate HOST line to your AmiTCP/IP database, and the ARexx file is
not executed.

1.12 PPP.guide/NODE_INST_AMITCPTZB

Additional installation for AmiTCP/IP 3.0 beta
==

   * Insert the line
         ppp dev=Devs:networks/ppp.device IPTYPE=33 NOARP P2P
     in the file AmiTCP:db/interfaces. Be careful to not insert that
     line at the end of the file, i.e. there has to be at least one
     more line in the file after the line you added.

   * Add the line
         WITH netdb-myhost-tmp
     to your AmiTCP:db/netdb file. REMOVE any lines starting with HOST
     (like HOST 1.2.3.4 name) from the files AmiTCP:db/hosts,
     AmiTCP:db/netdb or any other file that AmiTCP:db/netdb includes.
     IP numbers for PPP are usually dynamic, so they CANNOT be
     specified in the permanent AmiTCP/IP database.

   * If you DO NOT KNOW the host name of your local Amiga (including
     the complete domain name, like myamiga.cc.uni.edu) or if it is
     assigned to you dynamically for each connection (this is usually
     the case if your PPP host uses dummy names like
     dialup26.cc.uni.edu for each dialup line): Create a script
     file AmiTCP:bin/startamitcp that contains the following commands:
         online devs:networks/ppp.device 0
         PPPConfigAmiTCP30b 0
         run <nil: >nil: AmiTCP:AmiTCP
         SYS:Rexxc/WaitForPort AMITCP
         execute T:AmiTCP.tmp

   * Else (if you KNOW the host name of your local Amiga): Assuming the
     name is myamiga.cc.uni.edu: Create a script file
     AmiTCP:bin/startamitcp that contains the following commands:
         online devs:networks/ppp.device 0
         PPPConfigAmiTCP30b 0 myamiga.cc.uni.edu
         run <nil: >nil: AmiTCP:AmiTCP
         SYS:Rexxc/WaitForPort AMITCP
         wait 2 secs
         execute T:AmiTCP.tmp

   A word of explanation here: If you DO NOT specify a host name with
the PPPConfigAmiTCP30b command the resolve command is used to look up
your host name from a domain name server after AmiTCP/IP has been
started. This only works if your PPP service provider has actually

assigned a name to your PPP port, and if you have set up the IP address
of your domain name server correctly in the file AmiTCP:db/netdb-myhost.

   PPPConfigAmiTCP30b creates a temporary script file T:AmiTCP.tmp that
contains the following lines to configure AmiTCP/IP for PPP:
     AmiTCP:bin/ifconfig lo0 localhost
     AmiTCP:bin/ifconfig ppp<num> <localip> <remoteip>
     AmiTCP:bin/route add <localip> localhost
     AmiTCP:bin/route add default <remoteip>
     rx T:AmiTCP.rexx
   where <num>, <localip> and <remoteip> are dynamically replaced by
the correct values.

   These statements configure a minimum AmiTCP/IP environment using the
PPP link as the default link. If you want to use other network devices
in addition to PPP, either add the necessary configuration statements
to the file AmiTCP:bin/startamitcp, or modify and recompile the program
PPPConfigAmiTCP30b. The SAS/C source code is included in the directory
src/.

   The line rx T:AmiTCP.rexx is only added if your host name is
assigned to you dynamically. In this case PPPConfigAmiTCP30b creates a
temporary ARexx file that tells AmiTCP/IP about your host name and IP
number AFTER startup. Otherwise PPPConfigAmiTCP30b simply adds the
appropriate HOST line to your AmiTCP/IP database, and the ARexx file is
not executed.

1.13 PPP.guide/NODE_INST_AMITCPFZD

Additional installation for AmiTCP/IP 4.0 demo
==

   During the installation of AmiTCP/IP 4.0demo the installation program
asks you several questions. Answer as follows:

   * Choose the network interface type "PPP".

   * Choose the network interface to be used as "PPP".

   * If you have a permanent (i.e. non-dynamic) IP address assigned to
     your Amiga, enter it as the "default IP address". Otherwise enter
     "0.0.0.0".

   * If the IP address of your PPP host is constant and known to you,
     enter it as the "destination address". Otherwise enter "0.0.0.0".

   * Enter an empty string for the netmask.

   * Enter an empty string for the default gateway.

   After that you have to make several changes to the configuration that
the Installer created.

   Changes to the "AmiTCP:bin/startnet" script:

    * Delete the first few lines that start with
      ".key",".bra",".ket",".def".

    * Add the line
            online devs:networks/ppp.device 0
      at the beginning of the file.

    * Change the line starting with "AmiTCP:bin/ifconfig ppp0 ..." to
            AmiTCP:bin/ifconfig ppp0 $ppp0iplocal $ppp0ipremote

    * Change the line starting with "AmiTCP:bin/route add ..." to
            AmiTCP:bin/route add $ppp0iplocal localhost

    * After that line, add the line
            AmiTCP:bin/route add default $ppp0ipremote

    Changes to the "AmiTCP:bin/stopnet" script:

    * Add the line
            offline ppp.device 0
      at the end of the file

    Changes to the "AmiTCP:db/interfaces" file:

    * Insert the line
            ppp0 DEV=DEVS:networks/ppp.device UNIT=0 IPTYPE=33 NOARP P2P
      somewhere in the file.

1.14 PPP.guide/NODE_INST_AMITCPFOC

                 Additional installation for AmiTCP/IP 4.1 commercial
==

    During the Installation of AmiTCP/IP 4.1, choose the configuration
style "PPP configuration".

    Then answer the questions about your serial device driver (usually
"serial.device"), its unit number (usually 0) and the baud rate. This
baud rate must match the baud rate between computer and modem, NOT
necessarily the baud rate of your telephone line.

    When asked about the MTU you should keep the value at the default of
1500 at this time, because this is the most compatible setting. You
can reconfigure later for a smaller number to improve efficiency.

    Next if you want to use PPP's builtin dialer, enter the file name of
the dial script you created earlier. Otherwise enter an empty string.

    At the next option menu you can choose to enable or disable

Carrier Detect
     (see
                 Carrier detect

```
                          )

Hardware-handshake (CTS/RTS)
      (see
                          7-wire handshaking
                          )

no EOF-mode
      (see
                          Diable serial device EOF mode
                          )

Use ODU
      (see
                          Use OwnDevUnit.library
                          )

shared mode
      (see
                          Open the serial device in shared mode
                          )
```

   At this time it is recommended you only enable "Hardware-handshake"
and perhaps "CD", if you have a proper modem cable that connects the
"CD" signal.

   After the AmiTCP installation is finished you should make the
following change to the 'AmiTCP:bin/stopnet' script:

   * Add the line
         offline ppp.device 0
     at the end of the file

   Finally, type "version AmiTCP:bin/bootpconfig" to check the version
number of the program "bootpconfig". If it is 1.3, you need to upgrade
to version 1.5: Get the file "AmiTCP/bpc15.lha" from the ftp-site
"kampi.hut.fi", unpack the archive and copy the file "bootpconfig" to
"AmiTCP:bin". If you do not have ftp-access please contact me at
"kruse@cs.ucf.edu" and I will send you the archive by e-mail.

1.15 PPP.guide/NODE_INST_ASTTF

```
Additional installation for AS-225
==================================
```

   Insert the line
     devs:networks/ppp.device 0 33 0
   to your sana2_devs file. Please check the AS-225 documentation for
the exact filename and file format.

   Sorry, there are no more installation instructions for AS-225
available at this time. Please follow the general AS-225 installation
instructions in the AS-225 documentation.

1.16   PPP.guide/NODE_INST_ENVOY

Additional installation for Envoy
=================================

   During the Envoy installation Installer asks you for the names of
network devices you want to use. Answer those questions correctly to
enable PPP. One of the questions asks for the IP packet type used by
PPP. Please enter the number "33".

   Sorry, there are no more installation instructions for Envoy
available at this time. Please follow the general Envoy installation
instructions in the Envoy documentation.

1.17   PPP.guide/NODE_INST_ENLANDFS

Additional installation for EnlanDFS
====================================

   * Add the option "DNCP" to your "ppp0.config" file. That file should
     look like
          serial.device 0 57600 0.0.0.0 7WIRE DNCP

   * During the installation of EnlanDFS the Installer asks you for a
     device name and unit number. Type "ppp.device" and "0".

1.18   PPP.guide/NODE_CONFIGURATION

                  Configuration
****

   The configuration file (usually 'ppp0.config') can contain the
following options. All options marked with '(r)' are only functional in
the registered version of the device. However you can still specify
them in the configuration file of the unregistered device version – in
which case they are ignored.

   Note to users of AmiTCP/IP 4.1 (commercial): If you use the default
PPP installation, the configuration file 'ppp0.config' is automatically
generated by the 'startnet' script each time you start AmiTCP/IP. This
means that you should not add any options to the 'ppp0.config' file
(because they are overwritten later). Instead add any options to the
line starting with '.def S2OPT' in the 'startnet' script. You can also
use reconfigure AmiTCP by rerunning the 'Config_AmiTCP' utility to make
any changes.

Here is a complete list of all options in AmigaDOS ReadArgs format
(i.e. '/N' means the parameter is an integer, '/S' means the keyword is
a switch, '/K' means that the keyword must be given along with the
parameter):

   SERNAME,SERUNIT/N,SERBAUD/N,IPSTR,CD=CARRIERDETECT/S,
7WIRE/S,MAXFAIL/K/N,MAXTERM/K/N,MAXCONFIG/K/N,TIMEOUT/K/N,
VJCMODE/K/N,MTU/K/N,USEODU/S,SERBUF/K/N,BADXONXOFF/S,ACCM/K,
NOEOFMODE/S,ONLINE/S,NOVJC/S,NOACPC/S,SAVE/K/N,NOID/S,LOG/K,
EXERCISE/K/N,LQR/K/N,STARTUP/K,PAP/K,CHAP/K,FCS/K/N,DIALSCRIPT/K,
SHARED/S,REMOTEIP/K,IGNOREFCS/S,NOREQ/S,IPCP/S,DNCP/S

                    7WIRE

                                        7-wire handshaking

                    ACCM

                                         Asynch Control Character Map (r)

                    BADXONXOFF

                                     Xon/Xoff compatibility mode (r)

                    CD

                                          Carrier detect

                    CHAP

                                          CHAP password file name (r)

                    DIALSCRIPT

                                     Specify a dialer script

                    DNCP

                                         Enable DECnet protocol

                    EXERCISE

                                      Exercise link (r)

                    FCS

                                           Change checksum type (r)

                    IGNOREFCS

                                      Ignore incoming checksums

                    IPCP

                                         Enable TCP/IP protocol

                    IPSTR

                                         Local IP address

                    LOG

                                          Enable event logging

                    LQR

                                          Enable link quality reporting (r)

                    MAXCONFIG

```
                                        Maximum configuration counter (r)

                  MAXFAIL
                                          Maximum failure counter (r)

                  MAXTERM
                                          Maximum terminate counter (r)

                  MTU
                                            Maximum transfer unit (r)

                  NOACPC
                                          Disable Addr/Ctl/Protocol compression (r ↩
                                            )

                  NOEOFMODE
                                        Disable serial device EOF mode

                  NOID
                                            Disable initial ID string

                  NOREQ
                                            Suppress all requesters

                  NOVJC
                                          Disable Van Jacobson compression (r)

                  ONLINE
                                            Automatically go online

                  PAP
                                            PAP password file name (r)

                  REMOTEIP
                                        Set the remote IP address

                  SAVE
                                          Save intermediate startup packets

                  SERBAUD
                                          Serial device baud rate

                  SERBUF
                                            Serial device buffer size

                  SERNAME
                                          Serial device name

                  SERUNIT
                                          Serial device unit

                  SHARED
                                          Open the serial device in shared mode

                  STARTUP
                                          Send startup string
```

                    TIMEOUT

                                              Timeout duration (r)

                    USEODU

                                               Use OwnDevUnit.library

                    VJCMODE

                                              Van Jacobson mode (r)

1.19 PPP.guide/NODE_7WIRE

7-wire handshaking
==================

     7WIRE

   tells ppp.device to enable 7-wire (RTS/CTS) handshaking mode in the
serial.device. This option is STRONGLY RECOMMENDED to avoid data loss.

   If you for some reason decide to use software handshaking (Xon/Xoff)
instead of hardware handshaking (RTS/CTS) and you use the registered
version of ppp.device, make sure your asynchronous control character
map includes Xon and Xoff, i.e. specify the option BADXONXOFF or
include bits 17 and 19 in your ACCM bitmap.

   The unregistered version of ppp.device always uses the full ACCM
bitmap, so you can use software handshaking without any further
preparation.

   Make sure that your modem is set to the same handshaking
configuration that you specify to ppp.device. If you use 7WIRE, enable
hardware handshaking (AT&K3 on my Supra-V32bis). If you do NOT use
7WIRE, enable Xon/Xoff handshaking (AT&K4 on my Supra-V32bis).

1.20 PPP.guide/NODE_ACCM

Asynch Control Character Map (r)
===============================

     ACCM=56789ABC

   tells ppp.device that your serial line cannot correctly transmit
some control characters. This option is similar to BADXONXOFF, but more
general. You must supply a 8-digit hexadecimal number that represents a
32-bit bitmask with a "1" for each character that your line cannot
transmit (i.e. that needs to be escaped).

   Example: ACCM=00F00008 has bits 3,20,21,22,23 set, i.e. characters
3,20,21,22,23 and the corresponding characters with bit 7 set

(131,148,149,150,151) will be escaped. The default is ACCM=00000000.

    Do not specify both options BADXONXOFF and ACCM, or the result will
be undefined.

    In the evaluation version ACCM is not negotiated and therefore fixed
at FFFFFFFF, i.e. all control characters are always escaped.

1.21 PPP.guide/NODE_BADXONXOFF

Xon/Xoff compatibility mode (r)
===============================

     BADXONXOFF

    tells ppp.device that your serial line does not transmit Xon/Xoff
(ASCII 0x11,0x13) correctly – usually because software handshaking is
used somewhere on the line.

    In this case ppp.device 'escapes' those characters (and the
corresponding characters with bit 7 set, i.e. 0x91, 0x93) as specified
in the HDLC standard.

    This option should only be used if necessary, because it can slow
down the connection.

    In the evaluation version ALL control characters are ALWAYS escaped,
i.e. the functionality of BADXONXOFF is always active.

1.22 PPP.guide/NODE_CD

Carrier detect
==============

     CD

    tells PPP.device to terminate the connection when the 'carrier
detect' line indicates that the modem has hung up. This requires a modem
cable that correctly transmits the CD signal and an appropriate modem
configuration. For my Supra–V32bis modem, the command to enable the CD
signal is AT&C1.

    Another effect of this option is: If you use the builtin dialer
together with the option CD, the dialer only executes the dial script
if the CD line of your modem is not already active. Otherwise ppp.device
renegotiates the connection and goes online immediately. This has the
advantage that after you reset your Amiga you can reestablish your PPP
connection just by making ppp.device online – without redialing. Of
course this only works if your modem is configured not to hang up the
line when your Amiga is rebooted.

1.23 PPP.guide/NODE_CHAP

                    CHAP password file name (r)
========================

     CHAP=filename

   If your host requires you to use CHAP authentication, first create a
CHAP password file as described in
                Installation
                . Then add this option
to your configuration file. <filename> is the complete path name of
your CHAP password file.

   If you allow remote logins into your Amiga, you should make sure
that your CHAP password file is not accessible to outside users.
Otherwise they might use your hostid and password to log into your
remote PPP host.

   In the evaluation version this option is not available.

                    CHAP file format
                                        Format of the CHAP password file

1.24 PPP.guide/NODE_CHAPFILE

                    CHAP password file

   If your host requires authentication using CHAP ('Challenge
Handshake Authentication Protocol'), you first have to ask your site
administrator for

   * <his_hostid>

   * your <password> (sometimes called <secret>)

   * <your_hostid>

   Your site administrator has to add these items to his database. If
you have a Unix account on the same system, your <password> might be
identical to the <password> of your Unix account, and <your_hostid>
might be identical to your login name. <his_hostid> usually is the
host name you are connecting to, but you should ask your site
administrator to find out the exact name. In one case <his_hostid> was
identical to the complete domain name of the host, like

pppserver.cs.foo.edu.

   Create a CHAP password file that contains a single line:
     his_hostid password your_hostid
   i.e. <his_hostid>, <password> and <your_hostid> each have to be
separated by a single space. Do not use quotes to separate
<his_hostid>, <password> and <your_hostid>. If <his_hostid>, <password>
or <your_hostid> contain any unusual characters, like spaces or
unprintable characters, replace them by $4A where 4A is the hex ASCII
code of that character. A single $ must be written as $$ (no quotes !).

   The file can contain multiple <his_hostid> <password> <your_hostid>
triples, each on a separate line. This can be useful if you use PPP to
connect to different hosts and need different <password>s. ppp.device
chooses the correct <his_hostid> <password> <your_hostid> triple by
matching the <his_hostid> value that is transmitted by the host during
CHAP authentication with one of the <his_hostid> <password>
<your_hostid> combinations. That's why it is important that you ask
your administrator for the exact spelling of <his_hostid>. Otherwise
PPP will not find the correct CHAP file entry and reject negotiation.

   The name and location of the CHAP password file does not matter. You
have to use the option CHAP=filename in your configuration file to tell
ppp.device about the CHAP filename. See
                Configuration
                 for details.

1.25 PPP.guide/NODE_DIALSCRIPT

                Use the builtin dialer
======================

     DIALSCRIPT=filename

   This option specifies a dialer script that will be executed before
the PPP negotiation starts. The script is executed whenever ppp.device
is made online while it is offline, with one exception: If you
specified the CD option ppp.device only executes the dialer script if
the CD line is not active, i.e. if the modem does not have a carrier.
The advantage of this is that if your Amiga has to be rebooted while
the PPP line is up, ppp.device will not try to execute the dial script
again if the CD line signals that the line is still up.

   The format of dial scripts is similar to the one used by
gwcslip.device, a SLIP implementation by Graham Walter, but has some
additional commands. For a complete description see
                The builtin dialer
                    .

1.26 PPP.guide/NODE_DNCP

```
Enable DECnet protocol
======================
```

     DNCP

   Usually ppp.device is configured to establish a TCP/IP connection.
However if you want to use ppp.device for use with DECnet instead of
TCP/IP, you have to specify the option "DNCP". This disables TCP/IP and
enables DECnet instead.

   If you want to use both TCP/IP and DECnet over the same ppp line you
have to add both of the options "DNCP" and "IPCP".

1.27 PPP.guide/NODE_EXERCISE

```
                  Exercise link (r)
=================
```

     EXERCISE=k

   If this option is specified the device sends a "DiscardRequest",
i.e. a null message, whenever the SEND line has been idle for at least
<k> seconds. This can be used to prevent the other side from hanging up
the line due to inactivity. The request is not echoed by the other side
and should therefore not have any impact on the overall performance.

   Example: If you want to exercise the link after 1 minute of
inactivity, specify EXERCISE=60.

   Please see
               Inactivity Timeout
                for a more detailed discussion of this
topic.

   In the evaluation version this option is not available.

1.28 PPP.guide/NODE_FCS

```
Change checksum size (r)
========================
```

     FCS=size

   This option can be used to negotiate a different checksum size for
the PPP link. Valid values for <size> are 0, 16 and 32. The default is
16.

Using an FCS size of 0 instead of 16 saves 2 bytes for most PPP
packets and can thus improve performance, but eliminates the extra
safety that the FCS check provides. On the other hand using an FCS size
of 32 increases safety, but reduces performance.

A value of 32 probably only makes sense on very noisy lines that are
not error-corrected.

A VALUE OF 0 SHOULD ONLY BE USED ON ERROR-CORRECTED LINES, E.G.
MODEM LINES THAT USE MNP OR V42.

In the evaluation version this option is not available.

1.29 PPP.guide/NODE_IGNOREFCS

Ignore incoming checksums
=========================

    IGNOREFCS

tells PPP to ignore the checksums of incoming PPP packets after PPP
has been configured and has come online.

Normally PPP silently rejects all packets that have a bad checksum
(and relies on TCP-retries), as required by PPP specifications. However
some PPP terminal servers sometimes calculate incorrect checksums. The
option "IGNOREFCS" can be used to circumvent this bug.

Note that "IGNOREFCS" only becomes active after PPP has come online,
i.e. during the PPP configuration all checksums are always checked.
This is necessary to make sure that garbage left in the serial device
buffer is not incorrectly interpreted as a PPP packet.

Warning: You should only use this option on error-corrected (V.42 or
MNP) lines, and only if "PPPStats" shows you that you get many checksum
errors. This option is meant as a workaround for bugs in some terminal
servers (apparently some Annex models), NOT as a means to increase
performance in any way.

1.30 PPP.guide/NODE_IPCP

Enable TCP/IP protocol
======================

    IPCP

Usually ppp.device is configured to establish a TCP/IP connection as
default. You do not normally need the option "IPCP" to enable TCP/IP.

However if you want to use DECnet you can override this default by

specifying "DNCP". This enables DECnet and disables TCP/IP.

   If you want to use both TCP/IP and DECnet over the same ppp line you
have to add both of the options "DNCP" and "IPCP". This is the only
case where you need the option "IPCP".

1.31 PPP.guide/NODE_IPSTR

Local IP address
================

     IPSTR=123.45.67.89

   specifies your Amiga's local IP address if you know it. This is NOT
the IP address of the host you are connecting to.

   If you do not know your local IP address, or if you are dynamically
assigned a new IP address for each connection, you MUST specify 0.0.0.0
here.

   If this option is not specified the default is 0.0.0.0.

1.32 PPP.guide/NODE_LOG

Enable event logging
====================

     LOG=string

   enables event logging.

   To display the logged messages you must run the program "ppplog" in
its own shell window. The format of string can look like this (example):
     10S20G30
   The first integer (10 in the example) specifies the overall logging
level. This value must be given. This number can be followed by zero
or more instances of <character><integer> ("S20" and "G30" in the
example). These substrings change the logging level for specific types
of messages only. For those types this overrides the overall settings
given by the first parameter. Currently defined are:

G

     general messages

S

     messages related to SANA2 commands

L

     messages related to the LCP automaton

P

    messages related to the IPCP automaton

I

    messages related to IP packets

Q

    messages related to link quality monitoring

A

    messages related to the PAP automaton

C

    messages related to the CHAP automaton

  Logging level can be one of:

0

    No logging

10

    log errors only

20

    log errors, warnings and informational message from the peer

30

    log all of the above plus external events such as on/offline

40

    log all of the above plus all internal events (creates a large
    number of messages)

50

    log all of the above plus create a hexdump of all received and
    transmitted packets

  So the example sets logging for "S" to 20 and for "G" to 30, and for
all other types ("L", "P", "I", "Q", "A" and "C") to 10.

  Default: 0 for all types. When sending problem reports to the
author, please use LOG=50.

1.33 PPP.guide/NODE_LQR


```
Enable link quality reporting (r)
=================================


     LQR=k
```

  This option activates Link Quality Monitoring. It causes both sides
of the link to exchange packets every <k> seconds that tell each other
how many packets have been lost or damaged on the line. This
information can be used to determine the reasons of transmission

problems.

    In the evaluation version this option is not available.

1.34   PPP.guide/NODE_MAXCONFIG

Maximum configuration counter (r)
================================

     MAXCONFIG=10

    Changes the 'Max-Configure' value specified in the PPP RFCs. This
is the maximum number of Config-Req packets sent before ppp.device
gives up and assumes that the peer is unable to respond. Usually there
is no need to change this value. The default is 10.

    In the evaluation version of ppp.device this option is not
functional and the value is fixed at 10.

1.35   PPP.guide/NODE_MAXFAIL

Maximum failure counter (r)
===========================

     MAXFAIL=5

    Changes the 'Max-Failure" value specified in the PPP RFCs. This is
the maximum number of Config-Nak packets sent before ppp.device gives
up and assumes that the configuration will never converge. Usually
there is no need to change this value. The default is 5.

    In the evaluation version of ppp.device this option is not
functional and the value is fixed at 5.

1.36   PPP.guide/NODE_MAXTERM

Maximum terminate counter (r)
=============================

     MAXTERM=3

    Changes the 'Max-Terminate' value specified in the PPP RFCs. This
is the maximum number of Terminate-Req packets sent before ppp.device
gives up and assumes that the peer is unable to respond. Usually there
is no need to change this value. The default is 3.

    In the evaluation version of ppp.device this option is not

functional and the value is fixed at 3.

1.37 PPP.guide/NODE_MTU

```
Maximum transfer unit (r)
=========================
```

     MTU=1500

   changes the Maximum Transfer Unit.

   The Maximum Transfer Unit (MTU) is the maximum packet size that can
be transferred over the PPP link. The default is 1500.

   It is often useful to reduce this number to improve interactive
response time, particularly if you want to run FTP downloads and telnet
sessions in parallel.

   There are two ways to reduce this number:

   * MTU=123

     where 123 is the desired MTU value. This value MUST be less than
     or equal to 1500. This forces the connection to the specified
     value in both directions and causes ppp.device to give up if the
     peer does not support that value.

   * MTU=0

     causes ppp.device to wait for the peer to make a suggestion about
     the MTU value, and then use that value for both directions of the
     link. This is a good choice if the remote PPP host 'knows' which
     MTU is best.

     A technical note: This option is not completely RFC-compliant.
     (It is however to some degree similar to the obsolete "listen"
     state described in RFC 1171.)

     DO NOT USE 'MTU=0' ON BOTH ENDS OF THE SAME CONNECTION OR YOUR
     CONNECTION MAY APPEAR LOCKED UP AND WILL EVENTUALLY TIME OUT.

   Another technical note: There is a reason why this option cannot be
implemented on the Amiga exactly as specified in the PPP RFCs: The
SANA-II specs require MTU to be identical for both directions of a
link, whereas PPP negotiates the MTU values for both directions
separately and independently.

   In the evaluation version of ppp.device this option is not
functional and the MTU value is fixed at 1500.

1.38 PPP.guide/NODE_NOACPC

```
Disable Addr/Ctl/Protocol compression (r)
=========================================
```

     NOACPC

   disables Address/Control field compression and Protocol field
compression. This is probably only useful if the implementation of
those compression methods at your peer is buggy.

   In the evaluation version Address/Control field compression and
Protocol field compression are not implemented, so the option NOACPC is
always active.

1.39 PPP.guide/NODE_NOEOFMODE

```
Disable serial device EOF mode
==============================
```

     NOEOFMODE

   tells ppp.device NOT to use the serial.device EOF mode.

   Usually it is more efficient to use EOF-mode, particularly if the CD
switch is NOT used, because this reduces the number of IORequests and
thus the overhead.

   However some drivers of some multi-serial boards do not support
EOF-mode. As far as I know, one of those boards is the Commodore A-2232
board. If you want to use ppp.device with one of those boards you have
to specify this keyword.

1.40 PPP.guide/NODE_NOID

```
Disable initial ID string
=========================
```

     NOID

   Normally ppp.device sends an LCP-Identification to the remote host
before sending the first ConfigRequest.

   Specifying the 'NOID' switch suppresses this identification message
for compatibility to buggy PPP software. If you have problems
connecting to your host you might want to try this switch, even though
it should not be necessary for correctly implemented PPP hosts.

1.41 PPP.guide/NODE_NOREQ

```
Suppress all requesters
=======================
```

       NOREQ

    Usually ppp.device displays requesters if one of the configuration
files has an invalid format, if the underlying device cannot be opened,
or if the underlying device unit is locked.

    The 'NOREQ' switch suppresses these requesters, i.e. ppp.device
returns an error code to the calling program immediately.

    There is still one requester that cannot be suppressed: It is the
requester that appears when the configuration file 'ppp0.config' has an
invalid format. The option 'NOREQ' does not affect this requester,
because the requester indicates that the configuration file (including
the 'NOREQ' option) could not be parsed.

1.42 PPP.guide/NODE_NOVJC

```
Disable Van Jacobson compression (r)
====================================
```

       NOVJC

    disables Van Jacobson TCP compression. This is probably only useful
if the Van Jacobson implementation at your peer is buggy, or if you
have a VERY fast serial line and a slow CPU.

    In the evaluation version Van Jacobson TCP compression is not
implemented, so the option NOVJC is always active.

1.43 PPP.guide/NODE_ONLINE

```
Automatically go online
=======================
```

       ONLINE

    tells ppp.device to go online after initialization even if no
explicit online command is given. This option might be required for
Envoy, because the Envoy versions I have seen so far do not recognize
that a device is offline and keep flooding it with CMD_READ commands
:-(.

    For other protocol stacks (AmiTCP/IP, AS225) this option should not
be used.

1.44   PPP.guide/NODE_PAP

                    PAP password file name (r)
==========================

     PAP=filename,hostid

   If your host requires you to use PAP authentication, first create a
PAP password file as described in
                    Installation
                    . Then add this option to
your configuration file. <filename> is the complete path name of your
PAP password file. <hostid> is the host id that your remote host
administrator assigned to you. PPP.device uses <hostid> to look up the
corresponding password in your PAP password file. The format of the
<hostid> string is the same as that of <string> for the STARTUP option
(see
                    STARTUP
). Both arguments must be separated by a comma (',').

   If you allow remote logins into your Amiga, you should make sure
that your PAP password file is not accessible to outside users.
Otherwise they might use your hostid and password to log into your
remote PPP host.

   In the evaluation version this option is not available.

                    PAP file format
                                        Format of the PAP password file

1.45   PPP.guide/NODE_PAPFILE

                    PAP password file

   If your host requires authentication using PAP ('Password
Authentication Protocol'), you first have to ask your site
administrator for

   * your <hostid>

   * your <password>

   Your site administrator has to add these items to his database. If
you have a Unix account on the same system, <hostid> and <password>
MIGHT BE identical to the <username> and <password> of your Unix
account.

   Create a PAP password file that contains a single line:
     hostid password

i.e. <hostid> and <password> have to be separated by a single space.
Do not use quotes to separate <hostid> and <password>. If <hostid> or
<password> contain any unusual characters, like spaces or unprintable
characters, replace them by $4A where 4A is the hex ASCII code of that
character. A single $ must be written as $$ (no quotes !).

    The file can contain multiple <hostid> <password> pairs, each on a
separate line. This can be useful if you use PPP to connect to
different hosts and need different <hostid> <password> combination.
ppp.device chooses the correct <hostid> <password> pair by matching the
<hostid> value from the PAP option in your configuration files with one
of the <hostid> <password> combinations.

    The name and location of the PAP password file does not matter. You
have to use the option PAP=filename,hostid in your configuration file
to tell ppp.device about the PAP filename. See
                Configuration
                 for
details.

1.46   PPP.guide/NODE_REMOTEIP

Set the remote IP address
=========================

     REMOTEIP=123.45.67.89

    This option lets you specify the IP address of the remote PPP host.
The value specified here is only used if the remote PPP host asks for
it, i.e. if it tells ppp.device that its IP address is 0.0.0.0.

    If your remote PPP host is run by a PPP service provider it is
likely that the host knows its own address, so you do not have to use
this option. If you use this option anyway, it is ignored.

1.47   PPP.guide/NODE_SAVE

Save intermediate startup packets
=================================

     SAVE=10

    Some ppp hosts start transmitting packets IMMEDIATELY after the IP
connection has been established. The problem with this behavior is that
the local protocol stack (AmiTCP/IP or whatever) has not been
initialized at that time and has therefore not sent its first CMD_READ
request.

    This means that according to the SANA-II specifications those early

packets would be dropped – but although officially correct, in this
case this behavior is not desirable...

   To avoid this problem you can tell ppp.device to save at most <max>
incoming packets for CMD_READ requests that arrive later by using the
option 'SAVE=<max>'.

   This option should ONLY be used if PPP log messages tell you that
packets are dropped during initialization. Currently only IP packets
can be saved. Default: max=0.

   Warning: this option is completely untested right now and might
crash your machine. Unless your PPP host has a very unusual setup you
do not need this option and should not use it.

1.48 PPP.guide/NODE_SERBAUD

Serial device baud rate
=======================

     SERBAUD=57600

   specifies the baud rate for the serial port. This number MUST match
the baud rate you used in your terminal program or dialing script to
establish the dialup connection. If you use the builtin dialer this
speed can be set to any value that your modem supports.

   If this option is not specified the default is 9600 baud.

   Note: This number is usually NOT identical to the speed of your
phone line. Most modern modems use data compression (MNP-5 or V.42bis)
on the phone line, so the speed between computer and modem should be at
least twice as high as the physical line speed.

   Example: If your modem supports V32bis (up to 14400 Baud), your
computer-to-modem-speed should be 38400, 57600 or 115200 Baud. This is
the value you should use for SERBAUD and in your terminal program if
you use one to setup the connection. You should first read the manual
of your modem and make sure that your modem supports that particular
speed for its RS-232 line.

1.49 PPP.guide/NODE_SERBUF

Serial device buffer size
=========================

     SERBUF=16384

   changes the size of the serial.device receive buffer. The default is
16384 Bytes = 16 kB. You should not use very large values here, because

the PPP and TCP/IP protocols heavily depend on various timeouts. Large
serial buffers can defeat the purpose of timeouts, because they can
delay the delivery of packets. It is usually better to "drop" a packet
because of a full serial buffer, than to deliver a packet after a long
delay.

1.50 PPP.guide/NODE_SERNAME

Serial device name
==================

        SERNAME=serial.device

   specifies the name of your serial device. For third-party boards
this may for instance be gvpser.device.

   If this option is not specified the default is "serial.device".

1.51 PPP.guide/NODE_SERUNIT

Serial device unit
==================

        SERUNIT=0

   specifies the unit number for your serial device (physical port
number). This number has NOTHING to do with the PPP unit number. Unless
you want to use more than one PPP link at a time the PPP unit number
can always be 0. However the serial device unit number as specified
with this option must be set to the correct number of your serial port
(0 for the builtin Amiga serial port).

   If this option is not specified the default is 0.

1.52 PPP.guide/NODE_SHARED

Open the serial device in shared mode
=====================================

        SHARED

   If this option is used, ppp.device opens the serial device in shared
mode instead of exclusive mode. However keep in mind that only one
program can read data from a serial line at any time, so you cannot
have two programs (e.g. a terminal program and ppp.device) both read
data from the same serial line and expect this kind of setup to work.

The solution for this is to use only programs that support
OwnDevUnit.library (a freely distributable library that controls access
to device units) together with ppp.device on the same serial line, and
to use the USEODU option with ppp.device.

If SHARED is used in combination with USEODU, it may be possible to
have ppp.device "steal" a serial line from a getty-like
OwnDevUnit-aware program that owns that serial line.

1.53 PPP.guide/NODE_STARTUP

Send startup string
===================

    STARTUP=string

Note: This option has really been made obsolete by the new builtin
dialer. It is still there for backwards compatibility, but I recommend
you use the builtin dialer instead of STARTUP.

You can use this option to send <string> to the remote PPP host
immediately BEFORE the first configuration packets are exchanged.
Usually you do not need this option, because it is recommended that you
first establish the link manually, using the builtin dialer or by some
Rexx/Shell script and then startup ppp.device.

However I have been told that some PPP hosts have VERY short
timeouts that do not give you enough time to start PPP once you have
selected PPP mode on your host. If this is true for your setup, do not
send the final startup string (password etc.) in your script, but
specify it with this option instead. This may prevent your host from
timing out once the connection has been started.

Format of <string>: You can use any ASCII characters with one
exception: "$xx" specifies a character by its hex code, e.g. "$0D"
means <carriage return>. If you need the dollar sign ("$") in your
startup string, type "$$".

1.54 PPP.guide/NODE_TIMEOUT

Timeout duration (r)
====================

    TIMEOUT=3

Changes the 'Restart Timer' value specified in the PPP RFCs. This
is the timeout in number of seconds between subsequent configuration or
termination packets. If you have a very fast link you might want to
reduce this value to 2 or 1 to speed up the configuration process a
little bit.

In the evaluation version of ppp.device this option is not
functional and the value is fixed at 3.

1.55   PPP.guide/NODE_USEODU

Use OwnDevUnit.library
======================

    USEODU

  tells ppp.device to use OwnDevUnit.library (not included in this
archive) to negotiate access to the required serial.device unit.

  If SHARED is used in combination with USEODU, it may be possible to
have ppp.device "steal" a serial line from a getty-like
OwnDevUnit-aware program that owns that serial line.

1.56   PPP.guide/NODE_VJCMODE

Van Jacobson Mode (r)
=====================

    VJCMODE=2

  changes the default format of the Van Jacobson configuration option.
There are three possible values:

   * 0 to use the old RFC-1172 compliant format (4 bytes, type=0x37).

   * 1 to use an obsolete, intermediate format (4 bytes, type=0x2d).

   * 2 to use the new RFC-1332 compliant format (6 bytes, type=0x2d).
     This is the default.

  If you notice that Van Jacobson compression is not negotiated, or if
programs like telnet and ftp only work with Van Jacobson compression
disabled, you might want to try settings 0 or 1. They are used to make
ppp.device compatible to some older, buggy PPP software packages.

  This option only affects outgoing Config-Requests. For incoming
Config-Requests all formats (0, 1 and 2) are always automatically
accepted.

  In the evaluation version Van Jacobson compression is not
implemented at all, so this option is not functional.

1.57 PPP.guide/NODE_DIALING

```
The builtin dialer
******************
```

   The builtin dialer can be used to dial up a PPP line before PPP
option negotiation starts. To use the dialer create a text file that
contains a list of dialer commands and include the option
DIALSCRIPT=textfilename in your PPP configuration file.

   When PPP starts dialing, it opens a window to display error messages
and incoming data. If 'echo mode' is switched on, PPP also prints each
line of the dial script when it is executed.

   The dial script must contain a list of one or more of the following
commands. Each command must be given on a separate line. Lines that
start with a '#' as the first character are treated as comments (i.e.
are ignored).

   List of valid commands:

ECHO ON
     switches 'echo mode' on, i.e. each command in the dialer script is
     echoed when it is executed. Default: 'echo mode' is off.

ECHO OFF
     switches 'echo mode'off.

TIMEOUT ticks
     specifies the maximum amount of time (in ticks) that PPP is
     supposed to wait for a response from the serial line before a
     timeout occurs. Default: 500 (= 10 seconds).

SEND "textstring"
     sends textstring to the modem, and adds a 'Carriage Return'
     character, i.e. ASCII 13. textstring may not contain any control
     characters or double quotes. This command is usually used to send
     commands to the modem or to an intelligent terminal server.

SENDBIN "textstring"
     identical to the SEND command with two exception: No 'Carriage
     Return' is added at the end of the string, and the string may
     contain control characters: to send hex 0xab, include $AB in your
     string. This command is usually used if you need to send some
     "funny" caracters to a terminal server that the SEND command does
     not allow you to send.

WAIT "textstring"
     Waits until textstring has been received from the serial line, or
     until a timeout occurs, or until one of the strings specified by
     the ABORT or REDIAL commands has been received. This command is
     case-sensitive, i.e. if you wait for "test", this command will not
     recognize "Test" or "TEST" as equivalent. If you have to wait for
     a "Password" or "Username" prompt it is best to just wait for
     "assword" or "sername".

```
DELAY ticks
     waits for the specified number of ticks.

ABORT "abortstring1","abortstring2",...
     specifies which responses from the modem should abort the dial
     script and make PPP offline when WAITing for a string from the
     modem. Default: "BUSY", "NO CARRIER", "NO DIAL TONE" and "NO
     ANSWER".

REDIAL "redialstring1","redialstring2",...
     specifies which responses from the modem should cause PPP to
     redial. PPP does not have any knowledge about the logical state of
     your modem or of the connection, so "redialing" in this context
     just means that PPP executes the dial script again from the
     beginning. This is most likely what you want if the modem returns
     a "BUSY" or "NO CARRIER" message.  Default: (empty), i.e. PPP
     never redials.

REDIALDELAY ticks
     specifies the delay in ticks that PPP is supposed to wait before
     dialing again. Default: 1500 (= 30 seconds).

   Example dialscript:

     ECHO ON
     TIMEOUT 1500
     REDIAL "BUSY"
     SEND "AAT&D0DT5551234"
     WAIT "CONNECT"
     WAIT "sername"
     SEND "myusername"
     WAIT "assword"
     SEND "mypassword"
     WAIT "CS>"
     SEND "ppp"
     WAIT "MTU"
```

1.58 PPP.guide/NODE_INACTIVITY

```
Inactivity Timeout
******************
```

   First of all, the Amiga ppp.device NEVER causes an inactivity timeout
by itself. If you notice that the connection is terminated due to
inactivity it is always caused by the remote host.

   Many PPP hosts, PPP server programs, terminal servers etc. have
builtin functions that terminate the connection when the link has been
inactive for some amount of time. This is done in order to prevent
users from occupying valuable resources (modems :-)) when they do not
really use them. There are usually ways to avoid this effect by having
your Amiga "simulate" activity in certain intervals. Some of these
techniques are listed below.

Note: The following description is NOT intended as a hint how to
violate the regulations of your Internet access provider. Although
there may be legitimate reasons for simulating network events to
prevent inactivity timeouts, some Internet access providers have
regulations that consider those techniques "net abuse" and expressively
forbid them. In this case you MUST comply with those regulations, or
choose a different Internet access provider. Personally I do not
advocate or recommend any of the techniques given below and will not be
responsible if they violate any regulations of your Internet access
provider. You should get the permission of your Internet access
provider before using one of the following techniques.

Here are some strategies to prevent inactivity timeouts. The problem
is that different hosts have different ways of recognizing inactivity,
so not all strategies will work on all hosts – you have to experiment.

1. Use the option "EXERCISE=seconds" in your ppp0.config file. See
   above for details. This option sends "DiscardRequest" PPP packets
   to your PPP host every <seconds> seconds. This requires very little
   system resources and should not slow down your connection at all,
   because exercise packets are only transmitted when the link is
   otherwise idle. The disadvantage is that those exercise packets are
   swallowed by the PPP software and do not make it to the protocol
   stack of your PPP host. This means that you will fool terminal
   servers, but not intelligent PPP software. So if it is the PPP
   software on your host that causes the timeout (and not the
   terminal server), this option might not work. Also this option
   does not work in the evaluation version of ppp.device.

2. Another strategy (thanks to Stefan G. Berg for the suggestion):
   Use the "ping" command in delayed-loop-mode like this:
        ping -i seconds 123.45.67.89
   Where <seconds> is the number of seconds between consecutive polls
   and 123.45.67.89 must be replaced by the IP address of SOME
   machine reachable by your PPP host. This has the advantage that it
   creates real IP packets (ICMP, to be precise) and thus has a
   greater chance of fooling your host's PPP software. The
   disadvantange is that the request is echoed (slowing down long
   downloads) and that the remote host is "ping"ed even if you are
   using the link, i.e. this technique does not adapt to the load of
   your link.

3. Some PPP packages only take TCP (and sometimes UDP) packets into
   account when calculating inactivity time, not ICMP packets, so the
   previous approach might not always work. Here is yet another
   strategy: Use the program "letnet" in a delayed loop to create a
   TCP connection every once in a while, e.g. to get the remote system
   time. Example script:
        lab start
        letnet 123.45.67.89 37 >nil:
        wait 600 secs
        skip back start
   You have to replace the IP number and the time by values you need.
   The number "37" represents the port for the "time" service and
   should not be changed. This strategy has a slightly better chance
   of preventing inactivity timeouts, but requires even more
   bandwidth of your PPP link.

1.59   PPP.guide/NODE_RESTRICTIONS

```
Restrictions
************
```

   Here is a list of features that could, but need not be implemented
in a PPP driver and are not supported by the current PPP version yet.
First a list of those that I might implement in a future version of
ppp.device for registered users, given enough time.

   * A better dialer that supports features like dialing multiple
     numbers and has better modem control.

   * Reliable Transmission Mode (RFC 1663). For this I need a copy of
     the ISO 7776 and ISO 3309 documents. If you have access to one,
     please let me know. Right now I cannot implement this option.

   * Support for PPP over ISDN (RFC 1618), particularly some native
     support for raw PPP over a B channel. Since I do not have access
     to ISDN myself, I need your feedback on this. If ppp.device already
     works on your ISDN setup, please tell me. If not, please try to
     find out how your PPP host has configured PPP over ISDN. There
     seems to be more than one standard around...

   * Intelligent attempts to avoid loops during option negotiation. At
     this time only the MaxConfig counter is used to cancel the
     configuration negotiation after some time. There is no "per-option"
     loop detection yet.

   * "crossed connection" detection on out-of-sync RCA events

   * Real data compression. At this time the standard for PPP data
     compression has not been finalized yet, and it will probably take
     at least another couple of month until a draft standard becomes
     available. Then compression will probably be implemented using a
     general "Compression Control Protocol" (CCP), and one or more
     individual compression protocols. Newsflash: Because of patent
     claims on parts of the compression algorithms, PPP compression
     will probably not become available for quite a while...

   * A more "intelligent" logging mechanism. PPP should "know" about
     the most frequent IP packet types and port numbers and create log
     entries for them that are easier to read.

   Here is a list of features that are specified in various RFCs, but
are not supported by the current PPP version, and are not likely to be
implemented in the near future – or at all.

   However if someone really needs one of those features and agrees to
beta-test it for me, I might just implement it...

   * Using the local machine as a PPP host that recognizes incoming
     calls and dynamically assigns IP numbers to the remote (calling)

machine.

* Network Control Protocols other than IPCP and DNCP. At the moment
  IP and DECnet are the only supported protocol stacks. Support for
  other protocol stacks (ISO-OSI, AppleTalk, Novell IXP) is not
  implemented yet. If you know about a protocol stack for Amiga
  that implements one of those protocols and uses SANA-II calling
  conventions, please tell me !

* "Passive" option, i.e. option to stay on the line and wait for
  incoming configuration requests (might be useful for server
  operation).

* MTU values higher than 1500.

* RFC 1570 extensions, like "self describing padding", "callback",
  "compound frames", active "time remaining"

* MIB-II/SNMP support. This is a biggie and would require a lot of
  support in AmiTCP/IP to be useful.

* Options that still have draft status: reliable transmission mode,
  multilink mode, support for bridging, "Generic Authentication
  Protocol", "Arbitrary Handshake Authentication", Kerberos V4/V5
  authentication (anyone care to write a "des.library" and
  "kerberosv4.library" ?), "Callback Control Protocol".

Some of the implemented options are either untested, or have not
been tested thoroughly yet, because the software available to me did not
support them properly:

* SANA-II Rev 2.0 RejectProtocol callback for multiple protocol
  stacks

* NOEOFMODE

* SAVE option

* FCS alternatives

* robustness of PAP and CHAP, e.g. escape sequences in names,
  behavior after Nak etc.

* some commands of the dialer

1.60 PPP.guide/NODE_UTILITY

                 Utility programs
****

The following utility programs are included in the PPP archive:

                    PPPConfigAmiTCP30b

                                        Startup AmiTCP/IP V3.0beta.

                    PPPConfigAmiTCP23

                                        Startup AmiTCP/IP V2.3

                    PPPInfo

                                                Display information about a ←
                                            PPP unit

                    PPPStats

                                                Display statistics for a PPP ←
                                            unit

                    PPPLQR

                                            Display Link Quality Report

                    PPPLog

                                            Log PPP events

    These programs directly access the PPP internal data structures.

    As these structures are subject to change for each version, the
programs will usually work with exactly one version of ppp.device only.

    If you upgrade ppp.device please check for new versions of these
utility programs, too. If you attempt to start one of these programs
with a different version of ppp.device, an error message <program>
version and ppp.device version do not match will appear.

1.61 PPP.guide/NODE_PPPCONFIGAMITCPTZB

PPPConfigAmiTCP30b
==================

    This program is supposed to be started after the online command has
been executed for ppp.device. It reads the environment variables
created by ppp.device that contain the local and remote IP addresses
and creates a script file to start AmiTCP/IP V3.0beta.

    Usage: PPPConfigAmiTCP30b unit [hostname]

    <unit> must be given and specifies the PPP unit number.

    <hostname> is optional. If it is specified, PPPConfigAmiTCP30b
creates a HOST entry in the file AmiTCP:db/netdb-myhost-tmp. Otherwise
PPPConfigAmiTCP30b creates an ARexx file that calls the resolve program
to find out the local host name from the IP address and then sends an
ARexx command to AmiTCP/IP to update the AmiTCP/IP name database.

    The source code for this program is included in the archive.

1.62   PPP.guide/NODE_PPPCONFIGAMITCPTT

```
PPPConfigAmiTCP23
=================
```

   This program is identical to PPPConfigAmiTCP30b, except that it uses
the old interface names of AmiTCP/IP V2.3.

1.63   PPP.guide/NODE_PPPINFO

```
PPPInfo
=======
```

   This program displays some information about a PPP unit, such as the
configuration status and a list of active options.

   Usage: PPPInfo unit

   <unit> must be given and specifies the PPP unit number.

1.64   PPP.guide/NODE_PPPSTATS

```
PPPStats
========
```

   This program displays the statistics about a PPP unit, i.e. the
amount of received and transmitted data and a breakdown of all received
and transmitted packets by type.

   Usage: PPPStats unit

   <unit> must be given and specifies the PPP unit number.

1.65   PPP.guide/NODE_PPPLQR

```
PPPLQR
======
```

   This program displays the current Link Quality Monitoring statistics
about a PPP unit, i.e. the number of packets that were lost or damaged
on the link.

   Usage: PPPLQR unit

   <unit> must be given and specifies the PPP unit number.

The output of this program is either a link quality report or the message PPP unit %ld has not received any Link Quality Reports yet..

This message can appear

* if the number of link quality reports received so far is insufficient to build statistics.

* if the remote machine does not send any LQR packets (e.g. because it does not support LQR).

* if you have not specified option 'LQR' in your configuration file.

* if you are using the evaluation version of ppp.device.

1.66 PPP.guide/NODE_PPPLOG

PPPLog
======

This program prints log messages of all active PPP unit either to stdout, or to the builitin serial port.

Usage: PPPLog [SERIAL]

Usually PPPLog prints all log messages to stdout, which is the current console window unless you redirect it to a file using '>filename'.

If you specify the keyword SERIAL all output is sent to the builtin serial port instead by using 'kprintf'. This output can be redirected by other programs, like 'Sushi'.

Warning: If you specify the keyword SERIAL and do not have any equipment connected to your builtin serial port (and are not running any redirection program) your machine may crash !

1.67 PPP.guide/NODE_HISTORY

History
****

V1.30 (January 4th, 1995, 3rd public release)
        * Added support for AmiTCP/IP 4.0 (demo) and AmiTCP/IP 4.1 (commercial).

        * Added support for EnlanDFS (DECnet).

        * Fixed a bug in reading the configuration file.

            * Renamed "PPPConfigAmiTCP" to "PPPConfigAmiTCP30b" since it is
              only required for AmiTCP/IP 3.0b.

V1.25 (October 19th, 1994, limited release)
            * fixed a bug in IPCP configuration that (together with a buggy
              PPP implementation on the other side) could lead to infinite
              negotiation cycles.

V1.23 (October 13th, 1994, 2nd public release)
    no changes

V1.22 (October 6th, 1994, limited release)
            * option VJCMODE now works again

V1.21 (September 12th, 1994, limited release)
            * added new option IGNOREFCS to ignore the checksums of
              incoming packets

            * added new option NOREQ to suppress all requesters

V1.20 (August 30th, 1994, limited release)
            * fixed a bug in IP address negotiation

            * added new option REMOTEIP to actively set the IP address of
              the remote PPP host

            * zero PPP checksums are now silently accepted for compatibility
              with buggy PPP hosts...

V1.19 (August 18th, 1994, limited release)
            * fixed a bug that could crash the machine if a low-memory
              situation occured while ppp.device was online, but not opened.

            * added the dialer commands REDIAL and REDIALDELAY

            * removed character NUL from the character map in option
              BADXONXOFF because it does not appear to be necessary for
              correct software handshaking, but would slow down the
              connection significantly.

V1.18 (August 6th, 1994, limited release)
            * fixed a bug in PAP and CHAP.

            * fixed a bug in dialer related to abortargs handling

            * fixed a bug that could sometimes cause responses during
              option negotiation to deviate from specifications.

V1.17 (July 18th, 1994, limited release)
            * implemented SHARED keyword to be used in combination with ODU

            * implemented a dialer that is very similar to the one in Graham
              Walter's gwcslip.device. Thanks for the source code, Graham !

            * ppp.device now flushes the modem read buffer before going
              online. This is supposed to remove stale PPP packets from
              the input queue and prevent some race conditions in the LCP

             FSM when a connection is reconfigured after resetting the
             Amiga.

         * fixed a bug in PAP state machine that sometimes prevented
           proper reconfiguration after a reset.

V1.14 (May 16th, 1994, limited release)
         * fixed a bug related to incoming 32-bit FCS checksums

         * added PPPConfigAmiTCP23 to the distribution

         * yet another fix to the IPCP ADDRESSES problem

V1.11 (April 22th, 1994, limited release)
         * fixed another bug in the handling of obsolete IPCP ADDRESSES
           options.

V1.10 (April 21th, 1994, limited release)
         * added option FCS to the registered version to allow
           negotiation of different checksums.

         * added option CHAP (Challenge Handshake Authentication
           Protocol) to the registered version.

         * fixed the bug in PPPInfo that the number of V/J transmit
           slots was displayed incorrectly.

         * fixed a bug in the handling of obsolete IPCP ADDRESSES
           options.

         * PPPInfo now displays the FCS sizes.

V1.0 (April 14th, 1994, 1st public release)
         initial release

1.68 PPP.guide/NODE_ACKNOWLEDGEMENTS

Acknowledgements
****

    Thank you very much to

    * the beta testers Stefan G. Berg, Josh Goldsmith, Greg Jones, Luigi
      Mattera and Jukka Partanen.

    * Ignatio Souvatzis for some useful suggestions and his nice PPP FAQ
      list.

    * the AmiTCP/IP development group for their outstanding TCP/IP
      protocol stack AmiTCP/IP.

    * Graham Walter for sending me the source code and documentation of
      his gwcslip dialer.

* Kent Polk for information on configuring ppp.device for AS-225.

* all users who decide to register ppp.device.

* Kent Polk for information on configuring ppp.device for AS-225.

* all users who decide to register ppp.device.