Roadshow

TCP/IP networking for the Amiga computer

4 September 2023

by Olaf Barthel

Copyright © 2001-2023 by Olaf Barthel.

1 Introduction

Welcome to *Roadshow*, a TCP/IP stack for the Amiga. A TCP/IP stack allows you to connect to the Internet, access your e-mail, web pages, chat, etc. It can also help you access and exchange files within your local home network.

This documentation will try to walk you through the steps required to install Roadshow and make your Amiga part of the local network. A reference section will go into further detail, explaining what the individual parts of Roadshow do, and how they are used.

Please note that because the subject matter is a little on the complex side, it is recommended that you read at least the installation section. You might get lost if you skip it.

1.1 Feature set

Here is a short and by no means complete list of the peculiar features Roadshow has to offer:

- Compatible with the AmiTCP 4.0 API
- Amiga-ized configuration utilities rather than the terse and somewhat cryptic Unix legacy tools
- DHCP configuration for Ethernet devices; Zeroconf support for address allocation
- Support for Ethernet and PPP drivers.
- Complete SANA-IIR3 support, including DMA access feature
- Berkeley Packet Filter support
- IP packet filter/network address translation (NAT)
- `TCP:` device support
- Enhanced API to allow developers to take greater control of the TCP/IP stack, including features for monitoring and packet filtering
- Integration into the AmigaOS installation; configuration files go into `DEVS:` subdirectories, no special `Roadshow:` assignment necessary
- 'Simple' configuration: one single shell command can bring up the entire network; no unnecessarily complex configuration utilities
- Networking driver configuration concept similar to 'mount files', with drivers getting configured in the `User-Startup` script file
- Enhanced & cleaned-up software development kit, which includes the complete source code of all the configuration tools and sample programs
- Changes to global configuration files are tracked by file system notification; updates are made right after a change is detected
- Support for local Amiga Internet server programs (`inetd` style 'Internet superserver')
- Multicast support
- Support for ZeroConf automatic network interface IPv4 address assignment, in case no DHCP server is available
- Built-in support for international domain names using the ISO Latin 1 character encoding. Roadshow will transparently translate between domain names using the ISO Latin 1 character set encoding (umlauts, accented characters, etc.) and their internal PunyCode form.
- Built-in DNS cache, which reduces the number of server lookups.
- Complete localization of all tools and error messages

Roadshow also includes network device drivers for use with dial-up and ADSL networking. Here is a brief feature list which describes them:

- Separate drivers for dial-up networking over modem or ISDN (`ppp-serial.device`) and Ethernet/ADSL (`ppp-ethernet.device`) optimized for the respective connection type

- Special commands provided for connecting to the PPP server, which integrate with the Roadshow network setup
- Automatic configuration of routing and DNS servers
- Support for the password authentication protocol (PAP) and the challenge handshake authentication protocol (CHAP) using the MD5 hash function (RFC1334)
- Support for Van Jacobson IP header compression, with up to 16 slots, allowing for slot identifiers to be compressed (RFC1144, RFC1332)
- Protocol field compression and address and control field compression (RFC1331, RFC1661)
- Link quality monitoring (RFC1333)
- 16 bit frame checksums (RFC1331, RFC1661)
- The Microsoft extensions for the IPCP protocol which allow for the primary and secondary domain name server to be reported (RFC1877)
- Support for the Microsoft challenge authentication protocol extension known as MS-CHAP V1 (RFC2433)

1.2 Limitations

As of this writing the following limitations exist in Roadshow and the PPP/PPPoE drivers provided for it:

- All software requires Amiga operating system version 2.04 or higher to work. Older operating systems are unsupported.
- Roadshow is not compatible with the AmiTCP configuration utilities, such as the `ifconfig` or `route` commands.
- There is no built-in SSL/TSL library provided with Roadshow. But the freely available AmiSSL (see `https://github.com/jens-maus/amissl`) package works just fine.
- The configuration files and shell commands that ship with Roadshow are not entirely compatible with the respective files and commands you may be familiar with from the Unix domain.
- Roadshow is limited to the IP networking protocol, i.e. support for the ISO networking protocols present in the code Roadshow was derived from was removed.
- No data compression protocols are implemented in the PPP drivers. Trouble is, either the protocols are not sufficiently documented, and if they are, then the algorithms are either patented or the protocols are not generally available.
- No data encryption protocols are implemented in the PPP drivers.
- There is no multilink protocol support in the PPP drivers.
- There is no support for the MS-CHAP V2 authentication algorithm in the PPP drivers, as defined in RFC2759. From what my literature tells me, systems that support MS-CHAP V2 also ought to support CHAP with the MD5 Message Digest algorithm, so there should be no harm done. Also, as of this writing MS-CHAP V2 must be considered obsolete and a major security risk. It is unlikely that you will still encounter it.
- There is no support for 16 bit frame checksum alternatives (RFC1570) in the PPP drivers. It seems that there is little to be gained by implementing the zero or 32 bit checksum types; it complicates matters because some packets have to be sent twice, and because the end-to-end checksums implemented by the TCP/IP protocol already guarantee for a certain level of safety.

1.3 History

What did you do during those slow days between Christmas eve and New Year's eve 2000? I was reading Peter Bogdanovich's book *This is Orson Welles* and was just about starting to port a TCP/IP stack to the Amiga.

Why that? You may remember that the AmigaOS 3.5 update had shipped with an evaluation version of Holger Kruse's *Miami* integrated TCP/IP stack. A complete TCP/IP stack was to be included with the AmigaOS 3.9 update in the form of *AmiTCP Genesis*. However, the question of who actually owned *AmiTCP*, and whether it could be included legally with AmigaOS 3.9, led to complications.

At the time I started working on my own little pet project it was hard to tell whether any of the Amiga TCP/IP stacks was still commercially available, or was still supported by its developers.

I became curious as to how difficult it would be to port a TCP/IP stack to the Amiga. I eventually obtained a copy of the final BSD Unix release (the *4.4BSD Lite 2* distribution), extracted the networking code and started to have a little fun.

Approximately three weeks later I had the stack up and running. While my implementation shares a similar code base with AmiTCP (which is based upon the BSD Net/2 release – which is about 4-5 years older than the code I built upon), the API is just about the same and the memory management code is using the same tricks, this is where the similarities end. The remainder is a 'clean' port of the BSD networking code, starting from scratch.

I have transferred a few tiny code snippets from the *FreeBSD 4.2* TCP/IP stack to make this implementation less prone to LAND attacks, and I also applied all the code fixes suggested in *TCP/IP Illustrated, Volume 2* (plus fixes for buggy fixes). This doesn't make it a modern or bug-free TCP/IP stack, though.

As work progressed on making the TCP/IP stack more robust and useful I noticed that there was something missing. Designed to be modular, the TCP/IP stack did not include a built-in driver for the PPP dial-up networking protocol. At that time, there would exist only three PPP drivers for the Amiga: AmiPPP by Thomas Bickel and two different `ppp.device` implementations by Holger Kruse and Emmanuel Lesueur. None of these three was available for licensing.

There didn't seem to be much of an alternative to writing my own PPP driver from scratch. This turned out to be a challenging task, considering that just about everybody else out there simply adapts or ports the existing Australian National University PPP daemon. Yet this code was developed for a Unix host and AmigaOS is not necessarily an adequate environment for it to work in.

Getting the PPP driver written and tested took me much of the year 2001. Since I was going to get ADSL for my new home, work soon also included adapting the PPP driver to support the PPPoE protocol, which is a method for wrapping PPP data into Ethernet packets. There are other protocols which serve the same basic need (e.g. PPPoA, PPTP and L2TP) but which I found hard to support due to lack of adequate hardware. Also, the complexity of the protocols, in particular L2TP, is quite an obstacle.

During development of the PPP driver I found that the SANA-II specifications did not cater particularly well for dial-up networking drivers. At the time the standard was developed dial-up networking was not yet an important application. The typical network access was through Ethernet and maybe ArcNet, using a static configuration. The dynamic address configuration performed by dial-up networking drivers (PPP or SLIP) was beyond the scope of the original design. And the existing drivers so far had to resort to rather roundabout methods for configuring their parameters and communicating these to the application software (e.g. the TCP/IP stack). For example, a common practice was to store these parameters in environment variables which were then read by a script file. The script file would use the TCP/IP stack configuration utilities to translate the contents of the environment variables into network interface parameters.

Out of this work grew the proposed SANA-II, release 4 enhancements which both Roadshow and the PPP drivers support.

Due to how the PPP driver design evolved, it was comparatively easy to implement support for the PPPoE protocol. As a bonus, the implementation has very little overhead compared to other designs. Where some drivers have to parse and repackage plain PPP frames in PPPoE form, my PPPoE driver will generate and process PPPoE frames on the fly without any need for extra conversion.

The PPP and PPPoE drivers are designed to interface to a wrapper program. That program performs the dial-in operation and handles the driver configuration, registering it with the TCP/IP stack.

The basic feature set of the PPP drivers is comparable to the other available solutions. For example, Van Jacobson header compression is supported, as is MS-CHAPv1 authentication. Data compression, however, is not covered due to patent issues and general availability of the feature.

2 Legal stuff and acknowledgements

Since no part of the Roadshow TCP/IP stack implementation falls under the GNU General Public License I claim copyright protection for the modifications I made and reserve the right not to publish the source code. I wrote the PPP drivers `ppp-serial.device` and `ppp-ethernet.device` from scratch. Therefore *Roadshow*, `ppp-serial.device` and `ppp-ethernet.device` are Copyright © 2001-2023 by Olaf Barthel. All Rights Reserved.

If you bought a copy of this software, then you own the copy but not the original software. Your license to use the software allows you to make backup copies of it for personal use, but does not permit you to make, sell or distribute copies of the software.

You may use a single copy of the software on up to two different Amiga computers at the same time. If you want to use the software on more than two Amiga computers, you are required to buy another copy of Roadshow.

2.1 Third party software licenses and acknowledgements

2.1.1 TCP/IP stack and shell commands

The material which this TCP/IP stack is based upon falls under the BSD license, as reproduced below:

```
Copyright (c) 1983, 1990, 1993
     The Regents of the University of California.  All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software
   must display the following acknowledgement:
     This product includes software developed by the University of
     California, Berkeley and its contributors.
4. Neither the name of the University nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
```

2.1.2 IP filter

The IP filter code falls under the following license:

```
Copyright (C) 1993-2001 by Darren Reed.

The author accepts no responsibility for the use of this software and
provides it on an ``as is'' basis without express or implied warranty.
```

```
Redistribution and use, with or without modification, in source and binary
forms, are permitted provided that this notice is preserved in its entirety
and due credit is given to the original author and the contributors.

The licence and distribution terms for any publically available version or
derivative of this code cannot be changed. i.e. this code cannot simply be
copied, in part or in whole, and put under another distribution licence
[including the GNU Public Licence.]

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
```

2.1.3 IP header compression (PPP)

This implementation uses the IP header compression code written by Van Jacobson, for which the following notice applies:

```
Routines to compress and uncompress TCP packets (for transmission
over low speed serial lines).

Copyright © 1989, 1991, 1992, 1993 Regents of the University of
California. All rights reserved.

Redistribution and use in source and binary forms are permitted
provided that the above copyright notice and this paragraph are
duplicated in all such forms and that any documentation,
advertising materials, and other materials related to such
distribution and use acknowledge that the software was developed
by the University of California, Berkeley.  The name of the
University may not be used to endorse or promote products derived
from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.
```

2.1.4 DES (Data Encryption Standard)

The DES (Data Encryption Standard) encryption code used by the MS-CHAP V1 authentication code is Copyright © 1988, 1989, 1990, 1991, 1992 by Richard Outerbridge.

2.1.5 wget shell command

```
Copyright (C) 2005 Free Software Foundation, Inc.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

2.1.6 libpcap and tcpdump

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.
3. The names of the authors may not be used to endorse or promote
   products derived from this software without specific prior
   written permission.

THIS SOFTWARE IS PROVIDED ''AS IS'' AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

2.1.7 Acknowledgements and thanks

The TCP: handler design was reviewed and significantly rewritten by Jörg Strohmayer.

Over the course of more than 10 years, numerous contributors and reviewers helped to improve Roadshow and find bugs, in particular the AmigaOS4 beta test community.

Mike Mitchell was instrumental in finally getting Roadshow ready for publication, after a total of four previous attempts came to nothing.

My most profound thanks go to all of you!

2.2 Dedication

Roadshow is dedicated to the memory of my cat Felix, who died on 9 November 2012. He was barely five years old.

3 Requirements

There is no use pretending that installing and using networking software as powerful and complex as Roadshow will be straightforward and require no previous knowledge of networking.

So, I won't lie to you and suggest that you will be able to make use of Roadshow without having some basic knowledge of how IP networking actually works. This knowledge is essential for configuring your Amiga to become part of your home network, or the Internet.

As of this writing no graphical user interface is supplied with Roadshow, which means that you may have to use a text editor to add/change the respective configuration files. You also need to be proficient in using the Amiga shell.

3.1 Operating system requirements

The Roadshow TCP/IP stack and the `ppp-serial.device` and `ppp-ethernet.device` drivers require Amiga operating system 2.04 or higher to work. This includes Amiga operating system versions 3.0, 3.1, 3.5 and 3.9.

Operating systems which are highly compatible with these original Amiga operating system versions, such as MorphOS, are also expected to work with Roadshow. Note that this is, as of this writing, untested.

3.2 Computer hardware

The Roadshow TCP/IP stack and the `ppp-serial.device` and `ppp-ethernet.device` drivers work on any Amiga computer which matches the operating system requires described above, and which has at least 2 MBytes of memory installed.

The most basic hardware configuration with which the software was successfully tested was an *Amiga 600HD* running Amiga operating system version 3.1, equipped with 2 MBytes of memory, using a PCMCIA network adapter.

The more memory a computer has installed, and the faster it runs, the better.

3.3 Networking hardware and drivers

The Roadshow TCP/IP stack connects to the local network and/or the Internet by means of special devices and driver software.

These devices include Ethernet adapters, such as:

- Commodore A2065
- Village Tronic *Ariadne I*
- Village Tronic *Ariadne II*
- Any PCMCIA network adapter for which a working driver exists, e.g. `cnet.device` with PCMCIA *NE2000* hardware

Note that Roadshow assumes that the respective networking hardware is working correctly and that it has been properly connected to the local network.

Any network adapter hardware for which working drivers exist that follow the SANA-II standard should be usable with Roadshow.

Furthermore, software-only drivers which do not require any special hardware to be added to the Amiga in order to be useful, are supported, too. This includes drivers such as:

- `slip.device`
- `cslip.device`
- `liana.device`

- `plip.device`
- `ppp-serial.device`

As with the hardware drivers, as long as the respective network driver software follows the SANA-II standard it should be usable with Roadshow.

4 Installation

To begin installing Roadshow and the `ppp-serial.device` and `ppp-ethernet.device` drivers, double-click on the `Install_Roadshow` script icon, as provided with the software distribution.

The `Install_Roadshow` script will create new directories on your system partition, copy new files to it, and modify your `S:User-Startup` file.

You can, but need not restart your Amiga after the installation script has finished.

This concludes the easiest part of the installation. To make use of Roadshow you should read, understand and apply what is described in the following sections.

4.1 Configuring the network interfaces

Roadshow needs special device drivers which transport the network data between your Amiga, the local network and the Internet. A network interface configuration file is needed to make use of each such network device driver.

You can find the network device drivers in the drawer `DEVS:Networks`, which can be found in the `DEVS` drawer of your system boot volume.

Entering the command `Dir DEVS:Networks` may show the following list of files, perhaps even more than this collection:

```
a2065.device          ariadne.device
ariadneliana.device   slip.device
liana.device          plip.device
ppp-serial.device     ppp-ethernet.device
```

Caution: Not all network device drivers may be stored in the `DEVS:Networks` drawer.

If you just installed Roadshow, all the stock network configuration files which Roadshow shipped with will be stored under `Storage/NetInterfaces` on your system boot volume.

What kind of network device driver is required depends upon how your local network looks like, and what kind of network you want to access. Most importantly, you need a network configuration file which makes use of a device driver which is already present in your system and install this configuration file in the `DEVS:NetInterfaces` drawer.

Note: The stock network configuration files are almost intended for WLAN or Ethernet network devices, which make use of automatic address, route and name server configuration through the DHCP service (Dynamic Host Configuration Protocol) of your local network. These configuration files should mostly "work out of the box".

Roadshow ships with the '`ManageNetInterfaces`' command which will determine which network device drivers are availabe on your system and then gives you the option to install the corresponding stock network configuration files. You can do this manually (see Section 4.1.1 [Configuring for a cable modem or gateway router (ADSL)], page 13, which shows how to find the right file and adapt it, if needed), of course, but '`ManageNetInterfaces`' could save you time and effort.

Enter the following command in the shell:

```
ManageNetInterfaces
```

The '`ManageNetInterfaces`' command will proceed to gather the names of all device drivers which are currently available on your Amiga. These drivers will likely include network device drivers, such as those used in the network interface configuration files stored in the `SYS:Storage/NetInterfaces` and `DEVS:NetInterfaces` drawers. '`ManageNetInterfaces`' will look into every single such network interface configuration file and check if there is a matching network device driver for it.

Once '`ManageNetInterfaces`' is ready, it will show you which of these network interface configuration files may be installed:

```
ManageNetInterfaces
```

```
Available changes:

  In directory "SYS:Storage/NetInterfaces"
    Network interface file = Ariadne
    Network device driver  = ariadne.device
    Change                 = move to DEVS:NetInterfaces
                             (Network device driver available)

Use ManageNetInterfaces COMMIT INSTALL to perform these changes.
```

If you want these changes to be made, enter the command as suggested:

```
ManageNetInterfaces commit install
```

'`ManageNetInterfaces`' will show which changes it is going to make and then move the configuration files as needed:

```
ManagenetInterfaces commit install

Planned changes:

  In directory "SYS:Storage/NetInterfaces"
    Network interface file = Ariadne
    Network device driver  = ariadne.device
    Change                 = move to DEVS:NetInterfaces
                             (Network device driver available)

Move SYS:Storage/NetInterfaces/Ariadne to DEVS:NetInterfaces
All network interfaces files have been moved successfully.
```

At this point you should be able to start Roadshow and use it, by entering the following in the shell:

```
AddNetInterface Ariadne
```

4.1.1 Configuring for a cable modem or gateway router (ADSL)

For the remainder of this example it is assumed that the `ariadne.device` driver will be used. The `ariadne.device` is a device driver for the *Ariadne* Ethernet card.

You need to adapt or create a configuration file for the *Ariadne* network device. A sample file is provided in the file `SYS:Storage/NetInterfaces/Ariadne`. Copy this file to the `DEVS:NetInterfaces` drawer, e.g. by entering the following command in the shell:

```
Copy SYS:Storage/NetInterfaces/Ariadne#? DEVS:NetInterfaces
```

Open the copy in a text editor, e.g. by entering the following command in the shell:

```
Ed DEVS:NetInterfaces/Ariadne
```

The contents of this file should look as follows:

```
# $VER: Ariadne 1.2 (2010-11-27)
# Configuration for the Village Tronic "Ariadne" Ethernet card.

# The device name is mandatory
device=ariadne.device

# If not provided, unit number 0 will be used. You may
# have to change this if there are multiple cards of the
# same type installed in your machine, or if your network
# hardware supports several independent connections
#unit=0

# You must either pick a fixed (static) IPv4 address and
# a corresponding subnet mask, or request DHCP (dynamic)
# network address configuration.
# You can combine address/netmask/dhcp, which has the effect
# of asking the DHCP server to assign the requested IPv4
# address and subnet mask to this interface, if possible.
#address=192.168.0.1
#netmask=255.255.255.0
configure=dhcp

# If no DHCP server is present in your network, you can
# use automatic interface IPv4 address assignment through
# the ZeroConf protocol. Note that this will not set up
# default route and DNS servers for you, only the interface
# address is configured.
#configure=auto

# This variant of automatic IPv4 address assignment should
# be used in a wireless network instead of 'configure=auto'
#configure=fastauto

# You can enable diagnostic messages which can be helpful in
# tracking down configuration errors.
#debug=yes

# You can choose how much memory will be used when handling
# incoming and outgoing network traffic for this device.
# The default is to reserve 32 buffers of 1500 byte each, both
# inbound and outbound traffic. Larger values may provide
# better performance.
#iprequests=32
#writerequests=32

# For diagnostic and monitoring purposes it can be helpful to
# capture network traffic that flows through this interface.
# To enable this option, select one of filter=local,
# filter=ipandarp or filter=everything
#filter=local
```

```
#filter=ipandarp
#filter=everything

# The following options are specific to the Ariadne network
# hardware and work around limitations of the hardware
# design.
copymode=fast
requiresinitdelay=yes
```

The default configuration is set for automatic IP address, route and name server configuration through the DHCP protocol. If your networking card is working correctly, and is connected to a local network which has one or more routers present which respond to DHCP requests, then this basic setup should already provide you with working network access.

To give it a try, enter the following shell command:

```
AddNetInterface Ariadne
```

If everything goes well, the command should print something like the following text:

```
Interface "Ariadne" added.
Trying interface "Ariadne" configuration...
Interface "Ariadne" configured, address = 192.168.23.129, network mask = 255.255.255.0.
Added default route to 192.168.23.2.
Default domain name is "localdomain".
Added domain name server 192.168.23.2.
Interface "Ariadne" address 192.168.23.129 has been leased until 09/27/10 1:12 PM
   and will be renewed before it expires.
```

At this point you should be able to access the local network through Roadshow, and maybe even the Internet.

You can configure more network interfaces in exactly the same manner.

If things do not work out so well you might be seeing the `AddNetInterface` command print text like this:

```
Interface "Ariadne" added.
Trying interface "Ariadne" configuration...
Interface "Ariadne" configuration attempt timed out.
```

If this happens, then you will either need to pick a static IPv4 address for your Amiga, or the DHCP server in your local network is not responding/cannot respond to the requests your Amiga sent.

This example above describes the most basic network interface configuration that does not require any additional configuration work. You might be able to just copy an existing configuration file, edit the '`name`' and '`unit`' entries, and that should be it.

In some cases, however, you may have to configure each aspect the automatic DHCP configuration takes care of manually. This is described in the next following sections.

4.1.2 Configuring IP addresses, routing and name resolution

If your local network does not allow your network access to be automatically configured through a DHCP server, then you will have to configure the essential network settings manually. These settings consist of:

1. The IP address your Amiga will use
2. The IP address your Amiga will send data to which cannot be delivered to other computers in the local network (default route)
3. The IP addresses of servers which your Amiga will contact in order to associate IP addresses with names, and the other way round (name resolution)

Which choice of IP address you have for your Amiga, which default route should be used and which servers can be used for name resolution is either something you already know, or

information which can be provided to you by somebody who knows more about the network your Amiga is to be connected to than you do.

In any case, this information goes into three different configuration files.

The IP address your Amiga will use goes into the network interface configuration file which was already described in this document. For this example it is assumed that the name of this configuration file is `DEVS:NetInterfaces/Ariadne`.

Open this configuration file in a text editor, e.g. enter the following in the shell:

```
Ed DEVS:NetInterfaces/Ariadne
```

The section that needs editing should look like this:

```
# You must either pick a fixed (static) IPv4 address and
# a corresponding subnet mask, or request DHCP (dynamic)
# network address configuration.
# You can combine address/netmask/dhcp, which has the effect
# of asking the DHCP server to assign the requested IPv4
# address and subnet mask to this interface, if possible.
#address=192.168.0.1
#netmask=255.255.255.0
configure=dhcp
```

You will need to remove the "#" characters in front of the lines that contain the '`address`' and '`netmask`' information. You might also want to add a "#" character in front of the '`configure`' data.

To proceed, you need to know the IP address to assign to your Amiga and the corresponding subnet mask. In the example above, the IP address is '`192.168.0.1`' and the corresponding subnet mask is '`255.255.255.0`'. Note that you cannot just pick the address and subnet mask at will. You have to make sure that both are correct and that in particular the address is not currently being used by any other computer in your network.

Assuming that you picked the right address and subnet mask, and that they match the example configuration, your network interface configuration look something like this:

```
# You must either pick a fixed (static) IPv4 address and
# a corresponding subnet mask, or request DHCP (dynamic)
# network address configuration.
# You can combine address/netmask/dhcp, which has the effect
# of asking the DHCP server to assign the requested IPv4
# address and subnet mask to this interface, if possible.
address=192.168.0.1
netmask=255.255.255.0
#configure=dhcp
```

Save your changes back to disk.

The next file to be edited sets up the default route, which is often referred to as the default gateway. This is where Roadshow will send all the data intended for computers that are not part of your local network.

Open the file `DEVS:Internet/routes` in a text editor, e.g. enter the following command in the shell:

```
Ed DEVS:Internet/routes
```

The file should look like this:

```
# Routes which should be established when the protocol stack
# initializes itself. Each line corresponds to a route which should
# be established, very much like the parameters passed to the
# AddNetRoute command. The template is:
#
#  DST=DESTINATION/K,HOSTDST=HOSTDESTINATION/K,NETDST=NETDESTINATION/K,
#  VIA=GATEWAY/K,DEFAULT=DEFAULTGATEWAY/K
#
# You must either specify the default route, or specify a route with a
```

```
# destination address. Note that no duplicates are allowed, i.e. there
# can be only a single default route and the same route may not be
# established twice.
#default localhost
```

To proceed, you need to know the IP address of the computer which acts as the gateway router in your network. This IP address must be compatible with the IP address and subnet mask assigned to your Amiga, it cannot be picked at random.

Assuming that the IP address of the gateway router in your network is '`192.168.0.15`' you would add the following line to the the `routes` file:

```
default 192.168.0.15
```

Save your changes back to disk.

The final file to be edited sets up the IP addresses of the servers which Roadshow will contact in order to match IP addresses to host names, and the other way round. This is required in order to find the IP address corresponding to the name of a web server such as '`http://www.amiga.org`'.

Open the file `DEVS:Internet/name_resolution` in a text editor, e.g. enter the following command in the shell:

```
Ed DEVS:Internet/name_resolution
```

The file should look like this:

```
# This file is used by the name resolution process which associates
# raw IP addresses with domain names and the other way round. You
# can configure the addresses of the domain name system servers to
# use and how the lookup process will translate host names you provide
# before they are looked up.

# You can specify the IP addresses of up to three domain name system
# servers which will be queried in the order you provide here. Each
# line must start with 'nameserver' and then must be followed by
# an IP address.
#nameserver 192.168.0.1
#nameserver 192.168.0.2
#nameserver 192.168.0.3

# If a host name is to be looked up which has no trailing domain
# name attached, a default domain name can be appended to it
# prior to lookup
#domain name.com

# You can configure the list of host names for lookup. The search list
# is normally determined from the local domain name; by default, it
# begins with the local domain name, then successive parent domains that
# have at least two components in their names will be searched. This may
# be changed by listing the desired domain search path following the 'search'
# keyword with spaces or tabs separating the names. Up to six domain
# names may be listed.
#search my.first.domain my.second.domain

# The 'sortlist' parameter allows addresses returned by the name
# resolution process to be sorted. A sortlist is specified by IP address
# netmask pairs. The netmask is optional and defaults to the natural
# netmask of the net. The IP address and optional network pairs are separated
# by slashes. Up to 10 pairs may be specified.
#sortlist 130.155.160.0/255.255.240.0 130.155.0.0

# The dynamic configuration process for network interfaces (DHCP) may
# provide a set of domain name system servers. These dynamically-configured
# servers will always be consulted first before the statically-configured
# servers are queried. These statically-configured servers are set up in
# this file through the 'nameserver' configuration entries.
#
```

```
# You may prefer to use the statically-configured servers over the
# dynamically-configured servers, e.g. because they may be slower or
# unreliable. If this is the case, use the 'prefer static' option
# below, instead of 'prefer dynamic' (the default).
#prefer static
prefer dynamic
```

To proceed, you need to know the IP addresses of the name resolution servers, or at least one server. Assuming that the IP address of one such server is '`192.168.0.15`' you would add the following line to the the `name_resolution` file:

```
nameserver 192.168.0.15
```

If you want to use more than one server, add another line like the one above and replace the IP address.

Save your changes back to disk.

This series of edits prepared your for getting your Amiga connected to the local network and the Internet. As in the previous section which detailed how all this setup work could be done automatically through a DHCP server you can now try to enter the following command in the shell:

```
AddNetInterface Ariadne
```

Did it work?

4.1.3 Configuring for an ADSL modem

If you have a choice, use a gateway router to connect your Amiga to a broadband ADSL service. It is significantly more difficult to connect your Amiga directly to the same service, and less convenient, too.

But if you must, then you need to be prepared:

1. You need to use an Ethernet card to connect to the ADSL modem, which requires proper cabling, too. You have to know which device driver in the `DEVS:Networks` drawer is responsible for your Ethernet card.
2. You need to know the login and password issued to you by your Internet service provider.
3. You need to know which protocol your ADSL modem talks when it is connected to your Amiga. The only protocol which the drivers that are part of Roadshow understand is called PPPoE (PPP over Ethernet; "PPP" stands for "Point-to-point-protocol").

If you have everything ready, you can begin to set up the network interface file for the Roadshow PPPoE driver. A sample file is provided which you should copy to the right place, e.g. by entering the following command in the shell:

```
Copy SYS:Storage/NetInterfaces/PPPoE#? DEVS:NetInterfaces
```

You do not need to modify this file. It should work correctly as it is.

However, you do need to edit the configuration file which enables your Amiga to connect to the ADSL modem and set up a connection. A sample file is provided which you should copy before your edit it, e.g. by entering the following command in the shell:

```
Copy S:PPP-Configurations/PPPoE-Example S:PPP-Configurations/PPPoE
```

Open the copy in a text editor, e.g. by enterint the following command in the shell:

```
Ed S:PPP-Configurations/PPPoE
```

The contents of this file should look as follows:

```
# Example configuration for the "PPPoE" network interface, using
# the "ariadne.device" Ethernet driver.

# The name of the network interface, as known to Roadshow, is
# mandatory.
interface=PPPoE

# The name of the Ethernet driver is mandatory. Note that this
# must be an actual Ethernet network driver. Nothing else will do
device=ariadne.device

# This switch is necessary if you use Ethernet network drivers
# such as "ariadne.device" or "a2065.device". Without "raw=on"
# specified in this file the PPPoE driver will unable to talk
# to the ADSL modem. It works around bugs in the respective
# Ethernet device drivers.
raw=on

# This is where you enter the login and password provided to
# you by your Internet service provider. Replace the
# placeholders <mylogin> and <mypassword> with the login and
# password, respectively.
login=<mylogin>
password=<mypassword>
```

You may need to modify the lines which select the Ethernet driver name (‘`device=ariadne.device`’) and you will have to provide the login and password (‘`login=<mylogin>`’ and ‘`password=<mypassword>`’), respectively.

Save your changes back to disk.

To make your Amiga talk to the ADSL modem and set up a network connection, you first need to activate the PPPoE network interface, e.g. by entering the following command in the shell:

```
AddNetInterface PPPoE
```

If this worked out correctly, you can proceed to try setting up the network connection by starting the `ppp_connector` program. Enter the following command in the shell:

```
ppp_connector S:PPP-Configurations/PPPoE
```

If the ppp_connector program manages to set up the connection, the connection will persist until you close it. To close the connection, press the keys *[Ctrl]+C*, or use the shell `Break` command to stop the `ppp_connector` command.

4.1.4 Configuring for a serial port modem or ISDN adapter

This is one step more challenging than configuring your Amiga to talk to an ADSL modem. It is also the most old-fashioned way to connect to the Internet that might still work today.

If you must connect to the Internet through a serial port modem or ISDN adapter, then you need to be prepared:

1. You need to use a serial port modem, with the proper cabling, as connected to a working telephone line. Alternatively, you need an ISDN card, or an ISDN adapter you can connect your Amiga to as if it were a serial port modem.
2. You need to know which Amiga serial device driver to use, and which speed it safely operates at. The few ISDN cards available for the Amiga had their own special device drivers: you will need to know which driver is needed.
3. You need to know which commands are required to initialize your modem or ISDN card before it can be used.
4. You need to know which telephone number to dial in order for your modem or ISDN adapter to connect to the Internet.
5. You need to know the login and password issued to you by your Internet service provider.
6. You need to know which protocol your dial-up Internet service talks when it is connected to your Amiga. The only protocol which the drivers that are part of Roadshow understand is called PPP (Point-to-point-protocol over asynchronous serial lines).

If you have everything ready, you can begin to set up the network interface file for the Roadshow PPP driver. A sample file is provided which you should copy to the right place, e.g. by entering the following command in the shell:

```
Copy SYS:Storage/NetInterfaces/PPP#? DEVS:NetInterfaces
```

You should not need to modify this file. It ought work correctly as it is.

However, you do need to edit the configuration file which enables your Amiga to connect to the serial port modem or ISDN adapter and set up a connection. A sample file is provided which you should copy before your edit it, e.g. by entering the following command in the shell:

```
Copy S:PPP-Configurations/PPP-Example S:PPP-Configurations/PPP
```

Open the copy in a text editor, e.g. by enterint the following command in the shell:

```
Ed S:PPP-Configurations/PPP
```

The contents of this file should look as follows:

```
# Example configuration for the "PPP" network interface, using
# the "duart.device" serial port driver.

# The name of the network interface, as known to Roadshow, is
# mandatory.
interface=PPP

# The name of the serial port driver is mandatory.
device=duart.device

# The line speed to use when exchanging data with your serial
# port modem or ISDN adapter. This is given as the number of
# bits per second. This line is mandatory.
speed=115200

# This option enables hardware flow control operations when exchanging
# data with the serial port modem or ISDN adapter. This is a required
# option if the speed you selected is higher than 2400 bits per second.
rtscts=yes

# This option enables link testing: if the dial-up Internet service
```

```
# drops the connection, the PPP network interface will be shut down
# automatically.
checkcarrier=yes

# Your serial port modem or ISDN adapter may require an initialization
# command which will be sent before the Internet service is dialed.
# Always add the "\r" to the end of the command or it will not be
# properly sent to the modem!
#init=AT&FB8\r

# This is the actual command which is sent to your serial port modem
# or ISDN adapter, in order to connect to the dial-up Internet service.
# Always add the "\r" to the end of the command or it will not be
# properly sent to the modem!
dial=ATD0191011\r

# This is where you enter the login and password provided to
# you by your Internet service provider. Replace the
# placeholders <mylogin> and <mypassword> with the login and
# password, respectively.
login=<mylogin>
password=<mypassword>
```

You may need to modify the lines which select the serial port device driver name (‘`device=duart.device`’). You will most likely have to change the modem initialization command (‘`init=AT&FB8\r`’) and the dial command (‘`dial=ATD0191011\r`’).

Caution: If you need to use the star (‘`*`’) in a dial command, you should enclose the entire command in double quotes and instead of using just a single star, use two stars. The reason for the double star is a quirk in how the processing of the PPP configuration file works. Basically, a single star inside a double-quoted text has a special meaning and may not produce a ‘`*`’ for the dialing command. Only by making ‘`**`’ out of a ‘`*`’ will you be able to send the correct dial command to the modem.

You will have to provide the login and password (‘`login=<mylogin>`’ and ‘`password=<mypassword>`’), respectively.

Save your changes back to disk.

To make your Amiga talk to the serial port modem or ISDN adapter and set up a network connection, you first need to activate the PPP network interface, e.g. by entering the following command in the shell:

```
AddNetInterface PPP
```

If this worked out correctly, you can proceed to try setting up the network connection by starting the `ppp_dialer` program. Enter the following command in the shell:

```
ppp_dialer S:PPP-Configurations/PPP
```

If the ppp_dialer program manages to set up the connection, the connection will persist until you close it. To close the connection, press the keys *`[Ctrl]+C`*, or use the shell `Break` command to stop the `ppp_dialer` command.

Note that due to the complexity of this setup a lot of things can and probably will go wrong. I am afraid that if they do, you are mostly on your own here. Good luck!

4.2 Which files are installed, added or modified

The installation script will add libraries, network device drivers, configuration files and shell commands to your System partition. A few files may be modified. Here is a brief list of the files, where they go and what purposes they serve.

4.2.1 The "C" drawer (i.e. "C:")

Network configuration commands:

```
AddNetInterface
AddNetRoute
CheckRoadshowConfig
ConfigureNetInterface
DeleteNetRoute
ManageNetInterfaces
NetShutdown
RemoveNetInterface
RoadshowControl
```

Network information commands:

```
GetNetStatus
NetLogViewer
SampleNetSpeed
ShowNetStatus
ppp_sample
```

IP packet filter commands:

```
ipf
ipfstat
ipmon
ipnat
```

Diagnostic commands:

```
arp
ping
tcpdump
traceroute
```

File transfer commands:

```
ftp
wget
```

PPP and PPPoE connection/dial-up commands:

```
ppp_dialer
ppp_connector
```

Remote control and data exchange commands:

```
rsh
```

4.2.2 The "Devs" drawer (i.e. "DEVS:")

Network device drivers for PPP and PPPoE support in `DEVS:Networks`:

```
ppp-serial.device
ppp-ethernet.device
```

Network configuration files in `DEVS:Internet`

```
groups
hosts
name_resolution
networks
protocols
routes
rpc
servers
services
```

```
users
```

An empty drawer is added under `DEVS:NetInterfaces`.

4.2.3 The "Storage" drawer (i.e. "SYS:Storage")

Example configuration files for network interfaces, using a variety of network drivers are stored in the `SYS:Storage/NetInterfaces` drawer:

```
3c589
A2065
A314
AmigaNet
Ariadne
Ariadne_II
Asix
AteoNet
CNet
CNet16
Davicom
Discovery-II
IOBlix
MPC52xx
MediatorFast
MediatorNET
MosChip
Norway
PPP
PPPoE
Pegasus
Prism2
RTL8139
RTL8150
SunGEM
VIA-Rhine
X-Surf
X-Surf-100
X-Surf-500
plipbox
uaenet
```

Note that there may be many more files in this drawer than are mentioned above.

4.2.4 The "Libs" drawer (i.e. "LIBS:")

```
bsdsocket.library
usergroup.library
```

4.2.5 The "Locale" drawer (i.e. "LOCALE:")

Language files which replace the built-in text of the Roadshow libraries and shell commands:

```
Catalogs/deutsch/bsdsocket.catalog
Catalogs/deutsch/ppp-ethernet.catalog
Catalogs/deutsch/ppp-serial.catalog
Catalogs/deutsch/roadshow.catalog
Catalogs/deutsch/usergroup.catalog
Catalogs/polski/bsdsocket.catalog
Catalogs/polski/ppp-ethernet.catalog
Catalogs/polski/ppp-serial.catalog
Catalogs/polski/roadshow.catalog
Catalogs/polski/usergroup.catalog
```

4.2.6 The "S" drawer (i.e. "S:")

Example configuration files for PPP and PPPoE dial-up/ADSL network access:

```
PPP-Configurations/PPP-Example
```

```
PPP-Configurations/PPPoE-Example
```

Example configuration and script files for the IP packet filter:

```
Start-Firewall
Stop-Firewall
Check-Firewall-Rules
IPF/ipf.rules
IPF/ipnat.rules
IPF/examples/BASIC.NAT
IPF/examples/BASIC_1.FW
IPF/examples/BASIC_2.FW
IPF/examples/example.1
IPF/examples/example.10
IPF/examples/example.11
IPF/examples/example.12
IPF/examples/example.13
IPF/examples/example.2
IPF/examples/example.3
IPF/examples/example.4
IPF/examples/example.5
IPF/examples/example.6
IPF/examples/example.7
IPF/examples/example.8
IPF/examples/example.9
IPF/examples/example.sr
IPF/examples/firewall
IPF/examples/ftp-proxy
IPF/examples/ftppxy
IPF/examples/nat-setup
IPF/examples/nat.eg
IPF/examples/server
IPF/examples/tcpstate
```

The file `S:User-Startup` is modified to include the following lines:

```
;BEGIN Roadshow
If EXISTS S:Network-Startup
  Execute S:Network-Startup
EndIf
;END Roadshow
```

This will start the network only if a script file is present which does all the work necessary to perform the actual network setup. This script file is installed as `S:Network-Startup` and it looks as follows:

```
; $VER: Network-Startup 1.3 (22.11.2010)
FailAt 30
Run >NIL: NetLogViewer CX_POPUP=NO
AddNetInterface DEVS:NetInterfaces/~(#?.info) QUIET
```

The purpose of these lines is to start the network, using all network interfaces configured in the `DEVS:NetInterfaces` drawer, before Workbench opens. If any errors are detected during the network startup, the `NetLogViewer` command will capture them for later review.

5 Using the network

Roadshow allows your Amiga to become part of the local network and/or the Internet. The previous sections explained how the network drivers have to be configured in order to be used by Roadshow. This section will explain how Roadshow makes network access possible in the first place, how you can shut down Roadshow, and how you can find out about the current status of the network.

5.1 Startup

Even with no network interface configured and working, Roadshow will become active as soon as any program opens `bsdsocket.library` for the first time. At this stage programs can already communicate with each other just by exchanging data through the built-in *loopback* network interface.

As long as any program keeps `bsdsocket.library` open, the network access provided by Roadshow will remain operational.

If you want to go beyond the local computer, it is necessary to add and set up a network interface which is associated with an Ethernet card, a WiFi adapter, a dial-up line or an ADSL modem. How this may be done is described in the previous section.

By default a line added to the `S:User-Startup` file will take care of starting the network, adding and setting up network interfaces. This line reads as follows:

```
AddNetInterface DEVS:NetInterfaces/~(#?.info) QUIET
```

This command will collect the names of all network interface files stored in the `DEVS:NetInterfaces` drawer, sort them in alphabetical order and then proceed to add and configure each interface one at a time.

Note that if a network interface uses DHCP configuration, and no DHCP server is responding, it may take its time for the `AddNetInterface` command to abort the search for a DHCP server. If this happens, it will delay system startup.

Since the `AddNetInterface` command, as invoked by the `S:User-Startup` script file, will only take care of network interfaces with configuration files stored in the `DEVS:NetInterfaces` drawer, any other configuration files stored in the `SYS:Storage/NetInterfaces` drawer will not be used. If necessary, you will need to add these interfaces manually later.

By the time the `AddNetInterface` command has added and set up all network interfaces, your network access should be operational. Ideally, the whole operation should be finished quickly, and by the time the Workbench opens you should be able to access the network.

5.2 Status

By the time the Workbench opens your network access should either work, or there might be trouble. To find out what actually is configured and working, and which resources might be used, you can use three shell commands provided with Roadshow.

5.2.1 GetNetStatus

The GetNetStatus command performs simple configuration checks. Its main purpose is to be used in script files, where it can be used to check if specific configuration options are operational and configured, or not. It is also useful as a quick diagnostic tool which provides you with an overview of the current network operations.

To get a quick overview of your network configuration, enter the following in the shell:

```
GetNetStatus
```

If no network interfaces are currently configured, then the command might print a status report like the following:

```
bsdsocket.library 4.361 (27.8.2023) Roadshow 1.15 [Roadshow 4.361 (27.8.2023)]
No networking interfaces are available and configured.
No point-to-point networking interfaces are available and configured.
No broadcast networking interfaces are available and configured.
No name resolution servers are configured.
No routing information is configured.
The default route is not configured.
```

Having successfully added and set up a network interface, the same command would produce the following status report for an operational network:

```
bsdsocket.library 4.361 (27.8.2023) Roadshow 1.15 [Roadshow 4.361 (27.8.2023)]
Networking interfaces are available and configured.
No point-to-point networking interfaces are available and configured.
Broadcast networking interfaces are available and configured.
Name resolution servers are configured.
Routing information is configured.
The default route is configured.
```

In order to be useful, the network requires network interfaces to be configured and usable. This is what the following line in the status report above indicates:

```
Networking interfaces are available and configured.
```

Furthermore, name resolution services and routing must be configured and operational. This is what the following lines in the status report indicate:

```
Name resolution servers are configured.
Routing information is configured.
The default route is configured.
```

If all this data appears as part of the GetNetStatus report, then your network should be usable.

How the GetNetStatus command can be used in script files to test for specific network configuration options is described in detail in the reference section for the GetNetStatus command.

5.2.2 ShowNetStatus

The ShowNetStatus command provides you with detailed information about the current state of the Amiga's network configuration, and the resources currently in use.

To see a basic summary of the network configuration you can enter the following in the shell:

```
ShowNetStatus
```

This might produce a status report like the following:

```
Network status summary
Local host address         = 192.168.23.132 (on interface 'uaenet')
Default gateway address    = 192.168.23.2
Domain name system servers = 192.168.23.2
```

Compared to what the GetNetStatus command would have shown, this much briefer report tells you exactly which routing configuration is currently in use, and which network interface connects your Amiga to the local network.

To find out which network interfaces are currently configured and operational, you can enter the following in the shell:

```
ShowNetStatus interfaces
```

Which might produce a report like the following:

```
Name     MTU Type     Address        Received Sent Dropped Overruns Unknown Status
uaenet  1500 Ethernet 192.168.23.132       67   14       0        0       0 Up
```

This report lists the names of all currently configured network interfaces, as well giving a brief overview of the interface status. Note that information on the built-in *loopback* network interface is always omitted from this report.

More information about the network interfaces is available through the following command, which obtains a status report for the '`uaenet`' interface:

```
ShowNetStatus interface uaenet
```

This might produce a report like the following:

```
Interface "uaenet"
Device name                  = uaenet.device
Device unit number           = 0
Hardware address             = 00:0C:29:10:71:3E
Maximum transmission unit    = 1500 Bytes
Transmission speed           = 10000000 Bits/Second
Hardware type                = Ethernet
Packets sent                 = 15
Packets received             = 81
Packets dropped              = 0 (in = 0, out = 0)
Buffer overruns              = 0
Unknown packets              = 0
Address                      = 192.168.23.132
Network mask                 = 255.255.255.0
Number of read I/O requests  = 36 (maximum of 3 used at a time, 36 are still pending)
Number of write I/O requests = 32 (maximum of 2 used at a time, 0 are still pending)
Number of bytes received     = 7,517
Number of bytes sent         = 1,968
Transfer statistics (in/out) = DMA:0/0 Byte:81/15 Word:-/0
Address binding              = Dynamic
Address lease expires        = 10/03/10 12:11 PM
Link status                  = Up
```

To find out more about the name resolution server configuration, you would enter the following in the shell:

```
ShowNetStatus dns
```

This might produce a report like the following:

```
Address             Type
192.168.23.2        Dynamic

The default domain name is set to "localdomain".
```

Information about the routing configuration is available through the following command:

```
ShowNetStatus routes
```

Which might produce a report like the following:

```
Destination      Gateway          Attributes
(Default)        192.168.23.2     Up Gateway
127.0.0.1        127.0.0.1        Up Host
```

As before, compared with what the GetNetStatus command shows, this status report goes into more detail, listing the IP addresses in use, and how Roadshow uses them.

This was a brief overview of what the ShowNetStatus command can tell you about the current Amiga network configuration. More report information is available, though, and it is described in the reference section for the ShowNetStatus command.

5.2.3 CheckRoadshowConfig

If you are unsure where to start looking when Roadshow does not seem to work correctly, you might want to run a check over the configuration files which Roadshow currently makes use of. One or more of these files may not be active, contain typos or might even miss something.

Enter the following command in the shell to have all the Roadshow configuration files examined:

```
CheckRoadshowConfig Verbose
```

The '`Verbose`' switch will cause the CheckRoadshowConfig to print a short summary after having finished checking the files.

If your configuration files appear to be good shape, CheckRoadshowConfig will print the following message:

```
Your Roadshow configuration files seem to be in good order.
```

If problems were detected, you will see a list of the problems which were found, along with the names of the respective files and the line numbers where these appear. The last line then will then state that changes to the files may be necessary.

```
CheckRoadshowConfig: Network device driver name "moschipeth.device" should be changed t
CheckRoadshowConfig: Network device driver "moschipeth.device" not found; see line 7 of
You may have to update or repair your Roadshow configuration files.
```

5.3 Shutdown

Turning off or rebooting your Amiga should also shut down its network connection. But you might want to shut down the network in an orderly fashion instead, if that is possible.

This is what the `NetShutdown` command is for. It will attempt to tell every network program currently running to let go of the network resources and exit. When there is no more program using the network, the network interfaces will be deactivated, and finally the central `bsdsocket.library` controlling the network resources will be removed from memory.

To try and shut down the network you would enter the following in the shell:

```
NetShutdown
```

If everything goes well and the network could be shut down in an orderly fashion, you should see a message like the following:

```
Waiting for network to shut down... shutdown finished.
```

Note that the requests the `NetShutdown` command made to other network programs to let go of their network connections may have been denied or ignored. In this case the `NetShutdown` command will fail to accomplish its task and print a message like the following:

```
Waiting for network to shut down... timeout; network may shut down later.
```

If this message appears, then the network shutdown is still in progress and may conclude at a later time. You might want to try again later.

Note that the `NetShutdown` command will by default wait for up to 5 seconds for the shutdown to complete. If you wish to interrupt the shutdown attempt you can do so by pressing the *`[Ctrl]+C`* keys, or by using the `Break` shell command. In either case the `NetShutdown` command may print a message like the following:

```
Waiting for network to shut down... stopped waiting; network may shut down later.
```

Again, the shutdown command, once given, cannot be recalled. The shutdown may still conclude at a later time.

6 Troubleshooting

So far this documentation was written under the assumption that things tend to work out almost exactly as you intended them to, and that you understand what you are doing.

This section tries to give hints concerning known trouble spots with the Roadshow software, but it can only go so far. You still need to understand what you are doing, and basic knowledge of network operations is still a requirement.

You also need to be skeptical of the state of your network. Not every cable you are using can be assumed to be in good shape, or even plugged in where it should be. Your router may not actually be configured like you expect it to. Maybe you even made a typo when you entered that password. Or somebody might have deliberately broken your network.

6.1 Roadshow

6.1.1 Viewing error reports

By default Roadshow will start the network through the `AddNetInterface` command that is invoked as part of the `S:User-Startup` script file. If anything goes wrong at this stage, no error messages will be produced because the `AddNetInterface` command will keep mum about them.

If you suspect that the network startup may be in trouble, it can help to temporarily disable the `AddNetInterface` command invocation in your `S:User-Startup` file, and start the network manually. Here is how that command in `S:User-Startup` looks like:

```
AddNetInterface DEVS:NetInterfaces/~(#?.info) QUIET
```

To temporarily disable it, open the file `S:User-Startup` with a text editor (e.g. enter *`Ed S:User-Startup`* in the shell) and change the line to read as follows:

```
;AddNetInterface DEVS:NetInterfaces/~(#?.info) QUIET
```

Note the semicolon added in front of the `AddNetInterface` command. Save these changes back to disk and reboot your machine.

Once the Workbench is open, you can then try to start the network manually and observe what the `AddNetInterface` command said. Enter the following command in the shell:

```
AddNetInterface DEVS:NetInterfaces/~(#?.info)
```

Note that the `QUIET` option has been omitted: if anything noteworthy occurs, then the `AddNetInterface` command will no longer keep mum about it.

There is another tool to assist you in capturing error reports. It is started in the `S:User-Startup` script right before the `AddNetInterface` command is supposed to start the network. That tool is called `NetLogViewer`, and its purpose is to record and store network information and error messages for later review.

`NetLogViewer` will quietly run in the background until you tell it to open its message display window. This is done by either hitting a hot key combination, starting the program again, or through the Commodities `Exchange` command.

The default hot key combination used by `NetLogViewer` is *`[Shift]+[Alt]+F8`*

When you open the `NetLogViewer` window you may see a list of informational, warning or error messages. You can browse the message list, or you can use the `Project` menu to save the message list to disk.

No matter how you come by diagnostic and error messages, you should be able to save them for later review, and possibly to forward them to somebody who might make sense of them and assist you in fixing the problem at hand.

6.1.2 Network drivers

Only a few specific Ethernet solutions were produced specifically for the Amiga market. The respective hardware can be more than 20 years old by now (2012), and although it may still work correctly, the network device driver may be a problem.

The quality of the driver software produced for Amiga Ethernet solutions can vary greatly. If you are unlucky and stuck with a driver that is as old as the Ethernet card itself, chances are that it may not work robustly. Particularly tricky may be the driver for the A2065 Ethernet card, of which there are at least three different versions available.

I know that this is strange to suggest you go looking for a newer driver before you may have tried to use what you have to access the Internet.

Good places to start looking for the right drivers may be:

```
http://www.aminet.net
http://www.amiga-hardware.com
```

Some specific hardware has limitations which the driver software tries to work around. An example is the original *Ariadne* networking card. Because the `ariadne.device` driver by default moves each packet to be sent through an intermediate buffer, network performance can be noticeably limited.

To work around this workaround, Roadshow supports a configuration option which is by enabled in the *Ariadne* network configuration file. This option reads as '`copymode=fast`' and has the effect of bypassing the intermediate copying step which `ariadne.device` would otherwise end up performing.

There is another odd issue with the *Ariadne* networking card for which there is a workaround in the configuration file that ships with Roadshow. This workaround is enabled with the '`requiresinitdelay=yes`' option. It causes the driver initialization to be delayed by one second. Without this delay, the driver would end up dropping the first few incoming/outgoing network traffic packets.

6.1.3 Cabling

This may sound like an obvious bit of advice, but it is not: make sure that the Ethernet cables connecting your Amiga to the network are in good shape and that they are plugged in correctly before you try to start Roadshow.

This advice comes about because Roadshow is unable to detect whether or not a cable is connected to the Ethernet hardware you are using, due to network driver software limitations.

If you start Roadshow without a cable connected to the Ethernet card it should be using, then the automatic IP address assignment through DHCP will eventually fail, and Roadshow will not try again to make it work when you plug in the cabling again because it has no way to learn that the cable is now usable.

Should it be necessary to start Roadshow without correctly connected cabling, you may need to restart Roadshow manually later after the cabling has been correctly attached again: first shut down the network with the `NetShutdown` command, then repeat the `AddNetInterface` command as used in the `S:User-Startup` script file.

6.1.4 DHCP

The configuration files for the Ethernet network interfaces that ship with Roadshow default to use the DHCP (dynamic host configuration protocol) service to set up routing, name resolution and interface IP address.

The assumption is that you will have a working local DHCP server in your network, probably as part of a gateway router or cable modem.

But if this is not the case, then the network startup, as part of the setup performed by the `AddNetInterface` command invoked by the `S:User-Startup` script, will fail and eventually time out.

If that happens, you will notice that your Amiga appears to be stuck after a system restart, before the Workbench has even opened. It may take 10-20 seconds before the Workbench will eventually appear. At this point, however, the network interface that caused the delay will be unusable. You may have to restart the network manually.

To find out what happened, or failed to happen, you can enable the debug option for each network interface you want to monitor. For example, assuming that the *Ariadne* network interface may be giving you trouble, you would open the file `DEVS:NetInterfaces/Ariadne` in a text editor, e.g. by entering *`Ed DEVS:NetInterfaces/Ariadne`* in the shell.

In the file that is opened, look for the following line:

```
#debug=yes
```

Change that line to read as follows, or add another line with the following content:

```
debug=yes
```

Save your changes back to disk and restart your network. Messages about the progress of the DHCP exchange will be sent to the network debug console, or more likely, the `NetLogViewer` command.

So, how you fix the DHCP timeout that may be holding up your system startup procedure? I am afraid that there may be no straightforward solution, should it crop up and is not easily solved by having the cabling set up properly. As a last resort you may have to do without DHCP, pick a static IP address for your network interface and configure routing and name resolution manually. How this may be done is described in this documentation.

6.1.5 Network shutdown

Roadshow ships with a shell command which can be used to shut down the network in an orderly fashion, such as might be necessary before you can reconfigure the network (not everything is easily changed on the fly while the network is still in operation).

The problem with this command is that it may not always succeed. Unlike on a Unix system, it is not possible for an Amiga program to be forced to give up its network resources. This is what the `NetShutdown` command ultimatively may require, yet it can only work under the assumption that the programs currently using the network will comply with the request to shut down.

This is why the `NetShutdown` may not succeed at what it is supposed to do. Should it fail, you may have to close networking programs manually, or, as a last resort, just restart your Amiga.

In some cases it is, however, not the currently running networking software which may prevent `NetShutdown` from completing its task. Sometimes the very network device drivers themselves may not comply with the request to shut down. In such cases Roadshow may hang and not even the `NetShutdown` command will manage to complete its task. Restarting your system is then the only option available.

6.1.6 Tweaking network performance

Roadshow uses program code which was written at a time when Internet access from home involved a modem connected to the telephone line. The Internet was less robust than it is today, and data traveled more slowly across it. Also, computers were less powerful than today and had considerably less memory (even the Unix workstations which the TCP/IP stack was originally intended for).

Because of these historic conditions, Roadshow's internal default settings are made for a somewhat "conservative" network configuration. These settings should allow for best interoperability with a variety of network devices and servers, but they may be found wanting in this day and age,

because the means by which you connect to the Internet are both more robust and powerful than they used to be decades ago.

For example, you might want to change how much memory is devoted to handle incoming and outgoing traffic, and how TCP traffic is broken up into packets that travel across the network. These settings affect how well Roadshow uses the available resources, and will have an impact on data transmission speed.

Such changes are made using the `RoadshowControl` command (see Section 7.2.1 [Configuration], page 69, or consult the `RoadshowControl.doc` file).

The most basic tweaks you might want to try are the following, which should work well within the local network using Ethernet or Wi-Fi:

```
RoadshowControl set tcp.mssdflt = 1500
RoadshowControl set tcp.recvspace = 32768
```

In order to give these settings changes a try, enter the two commands above (in the shell) after you have started the network operations. These changes will lose their effects when you restart or turn off your Amiga.

If you like what you see, you can make these changes permanent:

```
RoadshowControl save set tcp.mssdflt = 1500
RoadshowControl save set tcp.recvspace = 32768
```

These changes affect how well Roadshow uses the available network bandwidth and memory and should make data reception faster than with the safe default settings.

Other changes are possible, too. The following were suggested during Roadshow's long test phase and work well with ADSL access to the Internet:

```
RoadshowControl set tcp.mssdflt = 1460
RoadshowControl set tcp.use_mssdflt_for_remote = 0
```

Because of how the TCP/IP stack goes about processing data received through the network interfaces, Roadshow will always use more memory to process the incoming data than should be necessary. You can change this behaviour so as to use only the minimum amount of memory necessary, at the expense of slightly higher processing effort:

```
RoadshowControl set if.receive.useclusters = 0
```

In this context the "cluster" is a 2048 byte sized chunk of memory into which the incoming data will be copied. If `if.receive.useclusters` is set to 1 (the default), then the data received will remain stored in that 2048 byte memory chunk until it has been processed. This may not be a good idea if your system has little free memory to spare. Setting `if.receive.useclusters` to 0 will move the inbound data to a new transport buffer which is just as small it needs to hold the data. This can save memory in the long run, but because it involves making a copy of the data just received, this may have a slight negative impact on performance. You might want to consider if this change is worth the trade-off.

Please consult the `RoadshowControl` documentation for a description of all the available settings and their respective effects.

6.2 The PPP/PPPoE drivers

Enable the logging feature the drivers offer. In the default operating mode, no error messages will ever be displayed, only the driver will refuse to go online. Error messages will be stored in the log file, or, if you requested that an output window is to be opened during the protocol establishment phase, displayed in the output window, too.

Check your configuration files for visible errors. One typo can ruin an entire day. The configuration file format is different for each kind of device, i.e. some options will not work for `ppp-serial.device`, while others will not work for `ppp-ethernet.device`. Pay close attention to the type of link layer device you pick. While `serial.device` might work for `ppp-serial.device`, it will not work for `ppp-ethernet.device`.

6.2.1 PPPoE with ppp-ethernet.device

Some Ethernet hardware drivers have problems sending PPPoE frames because they mistake the PPPoE frame types as hints to build IEEE 802.3 frames. This goes horribly wrong in that complete nonsense is committed to the wire.

If you see that several attempts are made to contact the PPPoE server on the network, and nothing comes forth, try to bring up the Ethernet hardware using your TCP/IP stack and ping a host on the network. If IP frames get through, it might be that your Ethernet hardware driver doesn't like the PPPoE frames. If you have the necessary hardware/software installed, try monitoring the Ethernet traffic (e.g. via `tcpdump` or a fancy tool like `Wireshark`) on your network when `ppp-ethernet.device` tries to open a session: if you see Ethernet frames of type '`8863`' making the rounds, which originated on your Amiga, then your Ethernet hardware driver is probably OK.

However, if you see strange LLC frames of type '`000D`', or something very much like that, then your Ethernet hardware driver might need some assistance. In that case, be sure to enable the '`RAW=ON`' option in the `ppp-ethernet.device` configuration file.

`ppp-ethernet.device` always tries to open the underlying Ethernet hardware driver in shared mode. This should work most of the time, but might fail if you have configured your TCP/IP stack to claim the driver in exclusive access mode, such as is necessary when you want a local Ethernet packet filter to have a look at every packet that flies by. In that case, you must disable the exclusive access option by disabling the filter (see the TCP/IP stack's documentation on interface configuration for more information).

I don't know yet how important it is to be able to specify the PPPoE service type to request from the server. The `ppp-ethernet.device` implementation attempts to verify that the service offered by the server matches the one it requests, but I'm not sure if this actually works the way it should. In any case, if it works, tell me about it, if it doesn't, tell me about it (and enclose a log file that shows what happens).

The current `ppp-ethernet.device` implementation is rather strict in assuming that if the underlying Ethernet driver reports that it's currently out of service, then no PPPoE session should be opened, or that the current PPPoE session should be closed. This was done that way because it was very convenient to implement, but it could be done better. Please advise.

I noticed that in my test configuration the PPPoE server requested that the PPP login and password should be transmitted in the open using the *Password Authentication Protocol* (PAP). Since that packet is transmitted on a broadcast medium (Ethernet), just about any packet sniffer can pick up the authentication information, which is a major security risk. The PPPoE server should use a safer protocol, such as the *Challenge Handshake Authentication Protocol* (CHAP) during the authentication phase. I recommend that you use the '`REJECTPAP=ON`' option to be on the safe side. If the server does not like your choice, you can always accept the default and omit the '`REJECTPAP`' option.

6.2.2 PPP with ppp-serial.device

Most modems don't talk to the computer unless data arrives with a certain speed. That means that you should pick a 'common' transmission speed in the configuration file, which is typically a multiple of 300, such as 19200, 38400, 57600, 115200, etc. 'Modern' modems can transmit so much data per second that you should always pick a transmission speed that is twice as large as the maximum your modem can deliver. For example, if your modem can send up to 57600 bits per second then you should pick 115200 as the transmission speed for the serial line driver. At those high speeds, the 7 wire hardware handshaking option (RTS/CTS) must be enabled, or you'll lose data.

The Amiga's built-in serial port hardware is notoriously unreliable when it comes to receive data at high speeds, such as anything beyond 19200 bits per second. Data will be dropped or corrupted upon reception. You might think that fancy new drivers for the built-in serial port hardware will allow for more reliable operation with higher transmission speeds, but I'd say that you shouldn't count on it to work. If you want to use higher speeds, use special serial port hardware, such as an old *MultiFaceCard*. If you don't have the necessary equipment handy, try lowering the transmission speed, even if that brings tears to your eyes.

The line PPP uses must not use software flow control using the *xON/xOFF* characters for signalling. If it does, you may have to change the '`ACCM`' configuration option to '`ACCM=000A0000`' or use the '`AT&K3`' modem command to enable hardware flow control.

If your modem initialization command doesn't echo anything back, it's pretty likely that you either didn't connect your modem to the computer, a cable is broken, or that the transmission speed is not correct. The latter is easiest to correct.

PPP only works with eight-bit-clean lines, i.e. your connection must use eight bits per data byte, no parity and one stop bit. If your connection strips parity bits or requires an uncommon number of stop bits, then you'll have a problem.

Line noise can distort the frames PPP transmits. Up to a point, you can help PPP filter out the noise by adding redundancy to the data transmitted. This is done by choosing an '`ACCM`' value of '`ACCM=FFFFFFFF`', which causes certain single bytes to be expanded into two byte sequences prior to transmission. Note that this will necessarily end up reducing throughput, but if you can't get any data sent at all, this may be just the price you want to pay. Don't expect any miracles, though.

If you need to use the star ('`*`') in a dial command, you should enclose the entire command in double quotes and instead of using just a single star, use two stars. The reason for the double star is a quirk in how the processing of the PPP configuration file works. Basically, a single star inside a double-quoted text has a special meaning and may not produce a '`*`' for the dialing command. Only by making '`**`' out of a '`*`' will you be able to send the correct dial command to the modem.

7 Reference

This section is intended to provide detailed descriptions of the different parts that make up Roadshow, including the PPP/PPPoE drivers: Which files are used by Roadshow, and what should be in them; which shell commands are provided, and what do they do?

Note that some of the descriptions in this section will refer to documentation files on disk, rather than just repeating their content. This is because the documentation being referenced is just too large to include on the spot.

7.1 Configuration files

7.1.1 Network interfaces in "DEVS:NetInterfaces" and "SYS:Storage/NetInterfaces"

Network interfaces are what make it possible for Roadshow to exchange data and information between your Amiga, its local network, and even between individual programs running on your Amiga.

The process of setting up a network interface is similar to how the `Mount` command works: it requires a configuration file which details which device driver should be used, and how it should be used, giving specific instructions.

The network interfaces are added through the `AddNetInterface` command; their configuration can be modified through the `ConfigureNetInterface` command; eventually, they can even be disable/shut down through the `RemoveNetInterface` command.

The `AddNetInterface` command will look for network interface configuration files in the `DEVS:NetInterfaces` and `SYS:Storage/NetInterfaces` drawers. The name of each such file stands for the name of a network interface. For historic reasons, network interface names cannot be longer than 15 characters. While you may be able to choose longer file names, the `AddNetInterface` command may choose to ignore anything but the first 15 characters.

Each network interface file consists of a number of lines of text, each of which can contain a comment or a configuration option.

Empty lines ignored when the respective text file is being processed. A line which begins either with the ";" or the "#" character is considered a comment and will be ignored, too.

A configuration option always starts with a keyword, such as '`device`', which is followed by a "=" character, which is followed by the option value. There can be any number of blank spaces preceding or following each of these.

Here is the list of options, as supported by the `AddNetInterface` command:

'`DEVICE/K`'
: Must be provided; the name of the SANA-II device driver. This should be the complete, fully qualified path to the driver. If no complete path is provided, the `DEVS:Networks` drawer will be checked. Thus, `DEVS:Networks/ariadne.device` is equivalent to `ariadne.device`.

'`UNIT/K/N`'
: Unit number of the device driver to open. The default is to use unit 0.

'`IPTYPE/K/N`'
: You can use this parameter to override the packet type the stack uses when sending IP packets; default is 2048 (for Ethernet hardware).

'`ARPTYPE/K/N`'
: You can use this parameter to override the packet type the stack uses when sending ARP packets. Default is 2054; this parameter only works with Ethernet hardware and should not be changed.

'IPREQUESTS/K/N'
: The number of IP read requests to allocate and queue for the SANA-II device driver to use. The default value is 32, larger values can improve performance, especially with fast device drivers.

'WRITEREQUESTS/K/N'
: The number of IP write requests to allocate and queue for the SANA-II device driver to use. The default value is 32, larger values can improve performance, especially with fast device drivers.

'ARPREQUESTS/K/N'
: The number of ARP read requests to allocate and queue for the SANA-II device driver to use. The default value is 4.

'DEBUG/K (possible parameters: YES or NO)'
: You can enable debug output for this interface (don't worry, you can always disable it later) to help in tracking down configuration problems. At this time of writing, the debug mode will, if enabled, produce information on the progress of the DHCP configuration process.

'POINTTOPOINT/K (possible parameters: YES or NO)'
: This indicates that the device is used for point to point connections. The stack automatically figures out whether the SANA-II device driver is of the point to point type, so you should not need to specify this option.

'MULTICAST/K (possible parameters: YES or NO)'
: This tells the stack that this device can handle multicast packets. 'YES' only works with Ethernet hardware (where it's enabled by default anyway).

'DOWNGOESOFFLINE/K (possible parameters: YES or NO)'
: This option is useful with point to point devices, like `ppp.device`. When specified, bringing the interface 'down' (via the `ConfigureNetInterface` program) or shutting down the stack will cause the associated SANA-II device driver to be switched offline (via the 'S2_OFFLINE' command).

'REPORTOFFLINE/K (possible parameters: YES or NO)'
: When a device is switched offline, you may want to know about it. This is helpful with SLIP/PPP connections which run over a serial link which accumulates costs while it is open. When the connection is broken and the device goes offline, you will receive a brief notification of what happened. However, if you tell the library itself to shut down, no notification that a device was switched offline will be shown.

'REQUIRESINITDELAY/K (possible parameters: YES or NO)'
: Some devices need a little time to settle after they have been opened or they will hickup and lose data after the first packet has been sent. The original *Ariadne* card is one such device. For these devices, the 'REQUIRESINITDELAY=YES' option will cause a delay of about a second before the first packet is sent.

  This option defaults to 'YES'.

'COPYMODE/K (possible parameters: SLOW or FAST)'
: This option is for chasing subtle bugs in the driver interface with cards like the original *Ariadne*. Cards like these do not support writing to the hardware transmit buffer in units other than 16 bits a piece. Default is 'SLOW', which is compatible with the *Ariadne*. But if you're feeling adventurous, try the 'FAST' option.

'FILTER/K (possible parameters: OFF, LOCAL, IPANDARP or EVERYTHING)'
: This option enables the use of the Berkeley packet filter for this particular interface. Possible choices for the key are:

'FILTER=OFF'
: Disables the filter.

'FILTER=LOCAL'
: Enables filtering on all IP and ARP packets that are intended for this particular interface. Packets intended for other interfaces or hosts are ignored.

'FILTER=IPANDARP'
: Enables filtering on all IP and ARP packets that happen to fly by this interface, no matter whether the packets are intended for it or not. This requires that the underlying network device driver is opened for exclusive access in so-called 'promiscuous' mode. This may not work if other clients (*Envoy*, *ACS*) need to keep the driver opened.

'FILTER=EVERYTHING'
: Identical to 'FILTER=IPANDARP', but will also filter all other kinds of packets that may show up.

  Default for this option is 'FILTER=LOCAL'. Note that by using this option you merely define what the filter mechanism can do and what it cannot do. The filter is not enabled when you add the interface.

'HARDWAREADDRESS/K'
: You can specify the hardware address (layer 2 address, MAC address) this interface should respond to when it is first added and configured. This usually works only once for each interface, which means that once an address has been chosen you have to stick with it until the system is rebooted. And it also means that the first program to configure the address will manage to make its choice stick.

  The hardware address must be given as six bytes in hexadecimal notation, separated by colon characters, like this:

```
HARDWAREADDRESS=00:60:30:00:11:22
```

  Take care, there are rules that apply to the choice of the hardware address, which means that you cannot simply pick a convenient number and get away with it. It is assumed that you will want to configure an IEEE 802.3 MAC address, which works for Ethernet hardware and is six bytes (48 bits) in size.

In addition to the purely static interface configuration information you can also tell the configuration program to do something about the interfaces once they have all been added. That's when the following configuration file parameters will be taken into account:

'ADDRESS/K'
: This configures the IP address of the interface. The parameter you supply should be an IP address in dotted-decimal notation ('192.168.0.1'). Don't pick a symbolic host name as the system may not yet be in a position to talk to name resolution server and translate the symbolic name.

  In place of the IP address you can also specify 'DHCP' (Dynamic Host Configuration Protocol). As the name suggests, this will start a configuration process involving the DHCP protocol which should eventually yield the right IP address for this host. Note that this configuration procedure only works for Ethernet hardware.

'ALIAS/K/M'
: In addition to the primary interface address you can assign several aliases to it. These must be specified in dotted-decimal notation ('192.168.0.1'). Alias addresses are added after the primary interface address has been configured.

'STATE/K' By default, interfaces whose addresses are configured will switch automatically to 'up' state, making it possible for the TCP/IP stack to use them for network I/O. You can override this by using the 'STATE=DOWN' switch.

'NETMASK/K'
This selects the subnet mask for the interface, which must be specified in dotted-decimal notation ('192.0.168.1').

In place of the subnet mask you can also specify 'DHCP' (Dynamic Host Configuration Protocol). As the name suggests, this will start a configuration process involving the DHCP protocol which should eventually yield the right subnet mask for this host. Note that this configuration procedure only works for Ethernet hardware.

'DESTINATION=DESTINATIONADDR/K'
The address of the point-to-point partner for this interface; must be specified in dotted-decimal notation ('192.168.0.1'). Only works for point-to-point connections, such as PPP.

'METRIC/K/N'
This configures the interface route metric value. Default is 0.

'MTU/K/N' You can limit the maximum transmission size used by the TCP/IP stack to push data through the interface. The interface driver will have its own ideas about the maximum transmission size. You can therefore only suggest a smaller value than the driver's preferred hardware MTU size.

'CONFIGURE/K (possible parameters: DHCP, AUTO or FASTAUTO)'
You can use DHCP configuration for this interface and protocol stack internals, namely the list of routers (and the default gateway) to use and the domain name servers. This option allows you to bring up the complete network configuration in one single step.

You can request that a particular IP address is assigned to this interface by the DHCP process by specifying 'CONFIGURE=DHCP' and your choice of 'ADDRESS=xxx.xxx.xxx.xxx'.

If your network has no DHCP server, you may choose 'CONFIGURE=AUTO' to use automatic IPv4 address selection, based upon a protocol called 'ZeroConf'. This protocol will select a currently unused address from a specially designated address range.

If you choose automatic configuration in a wireless network, you might want to use 'CONFIGURE=FASTAUTO' instead of 'CONFIGURE=AUTO'.

Note that only the 'CONFIGURE=DHCP' option will attempt to set up a default route and a set of DNS servers for you to use. The alternatives of 'CONFIGURE=FASTAUTO' and 'CONFIGURE=AUTO' are restricted to selecting the network interface IPv4 addresses.

'LEASE/K' This is a complex option which can be used to request how long an IP address should be bound to an interface, via the DHCP protocol. Several combinations of options are possible. Here is a short list:

'LEASE=300'
'LEASE=300seconds'
This requests a lease of exactly 300 seconds, or five minutes.

'LEASE=30min'
This requests a lease of 30 minutes.

'LEASE=2hours'
This requests a lease of 2 hours.

'LEASE=1day'
: This requests a lease of 1 day.

'LEASE=4weeks'
: This requests a lease of 4 weeks.

'LEASE=infinite'
: This requests that the IP address should be permanently bound.

Blank spaces between the numbers and the qualifiers are supported. The qualifiers are tested using substring matching, which means for example that '30 minutes' is the same as '30 min' and '30 m'.

Note that the requested lease time may be ignored by the DHCP server. After all, it is just a suggestion and not an order.

'ID/K'
: This option works along with the 'CONFIGURE=DHCP' process. It can be used to tell the DHCP server by which name the local host should be referred to. Some DHCP servers are on good terms with their local name resolution services and will add the name and the associated IP address to the local host database. The name you can supply here cannot be longer than 255 characters and must be at least 2 characters long. Keep it brief: not all DHCP servers have room for the whole 255 characters.

'DHCPUNICAST/K'
: Some DHCP servers may not be able to respond to requests for assigning IP addresses unless the responses are sent directly to the computer which sent the requests. In such cases you might want to use 'DHCPUNICAST=YES' option.

7.1.2 Roadshow configuration files in "DEVS:Internet"

Roadshow will read global configuration information from a number of files stored in the DEVS:Internet drawer. Most of these files have equivalents in the original Unix TCP/IP stack implementation, where they would be found in the /etc directory.

Please note that the layout and contents of the files used by Roadshow may not be identical to how the related, original Unix configuration file may have looked like.

7.1.2.1 groups

This purpose of this file is similar to the Unix /etc/group file, but uses a different format to store its information.

The contents of this file are used by software which needs to translate the names of groups a user may be a member of into numbers. A typical application would be the use of the Amiga NFS (network file system) client.

ppp_dialer processes the configuration file one line at a time. Each line starts with a keyword (e.g. 'interface'), which is then followed by the value to assign. A = between the keyword and the value is optional.

Caution: If the value you need to assign contains blank spaces, asterisks (*) or double quote characters ("), you will need to rewrite the value for it to be processed correctly.

- Blank spaces

  The value needs to be enclosed in double quotes. For example, Example text would have to be rewritten as "Example text". This is necessary because the processing of the value otherwise stops at the first blank space.

  Once you need to enclose the value in double quotes, you will need to check if you now need to rewrite it further.

- Asterisks

  If the value is enclosed in double quotes, each asterisk has to be replaced by two asterisk characters.

  For example, `"ATD*101"` would have to be rewritten as `"ATD**101"`. This is necessary because the asterisk character within a quoted value has a special meaning.

  If the character which directly follows the `*` is one of `e`, `E`, `n`, or `N`, then a control character (Esc or Line Feed, respectively) will be produced. This is likely not what you want.

  If the character which directly follows the `*` is `*`, a `*` will be produced. Hence, `**` produces `*`.

  If the character which directly follows the `*` is `"`, a `"` will be produced. Hence, `*"` produces `"`.

  If any other character follows the `*`, then that character will be ignored. This is likely not what you want.

- Double quotes

  If the value is enclosed in double quotes and the value itself contains double quotes, e.g. when used in a password string, each double quote has to be replaced by `*"` and the result then has to be enclosed in double quotes itself.

  For example, `#;$ "&.` contains blank spaces and also a double quote character which requires that it is rewritten and then enclosed in double quotes as well. Hence, `#;$ "&.` must be changed to `#;$ *"&.` and then enclosed in double quotes: `"#;$ *"&."`

You may want to limit the use of blank spaces, asterisks and double quote characters in your configuration file.

Each line of the `groups` file is read and processed according to the following template:

```
NAME/A,ID/A/N,USERS/M
```

Blank lines, or lines which start with the `"#"` character will be ignored.

The individual options have the following meanings:

'`NAME`' The name of the group which is defined here.

'`ID`' The number to associate with the group whose name is defined with the NAME option.

'`USERS`' A list of user names, as defined in the `DEVS:Internet/users` file, indicating that the user is a member of the group. Note that the group listed for a user in the `DEVS:Internet/users` file need not be repeated here.

7.1.2.2 hosts

This file is equivalent to the Unix `/etc/hosts` file. It contains information regarding the known hosts on the network. For each host a single line should be present with the following information:

```
official host name
Internet address
aliases
```

Items are separated by any number of blanks and/or tab characters. A `"#"` indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

Network addresses are specified in the conventional dotted decimal notation. Host names may contain any printable character other than a field delimiter, newline, or comment character.

Note that Roadshow will track changes to the `DEVS:Internet/hosts` file and automatically reread and process the file if necessary.

7.1.2.3 name_resolution

This file is equivalent to the Unix `/etc/resolv.conf` file. It contains information regarding to how name resolution should be performed. Name resolution in this context means turning IP addresses into human readable names and the other way round.

Each line of the file defines a configuration option. A "#" indicates the beginning of a comment. The different configuration options are:

'`nameserver`'

Internet address (in dot notation) of a name server that the resolver should query. Up to 10 name servers may be listed, one per keyword. If there are multiple servers, the resolver library queries them in the order listed. If no nameserver entries are present, the default is to use the name server on the local machine. (The algorithm used is to try a name server, and if the query times out, try the next, until out of name servers, then repeat trying all the name servers until a maximum number of retries are made).

'`domain`'

Local domain name. Most queries for names within this domain can use short names relative to the local domain. If no domain entry is present, the domain is determined from the local host name by looking up the IP address of the first usable network interface; the domain part is taken to be everything after the first "." character. Finally, if the host name does not contain a domain part, the root domain is assumed.

'`search`'

Search list for host-name lookup. The search list is nor- mally determined from the local domain name; by default, it begins with the local domain name, then successive parent domains that have at least two components in their names. This may be changed by listing the desired domain search path following the search keyword with spaces or tabs separating the names. Most resolver queries will be attempted using each component of the search path in turn until a match is found. Note that this process may be slow and will generate a lot of network traffic if the servers for the listed domains are not local, and that queries will time out if no server is available for one of the domains.

'`prefer`'

The dynamic configuration process for network interfaces (DHCP) may provide a set of domain name system servers. These dynamically-configured servers will always be consulted first before the statically-configured servers are queried. These statically-configured servers are set up in this `DEVS:Internet/name_resolution` file through the "nameserver" configuration entries.

You may prefer to use the statically-configured servers over the dynamically-configured servers, e.g. because they may be slower or unreliable. If this is the case, use the '`prefer static`' option.

The search list is currently limited to six domains with a total of 256 characters.

The domain and search keywords are mutually exclusive. If more than one instance of these keywords is present, the last instance will override.

The keyword and value must appear on a single line, and the keyword (e.g. nameserver) must start the line. The value follows the keyword, separated by white space.

Note that Roadshow will track changes to the `DEVS:Internet/name_resolution` file and automatically reread and process the file if necessary.

7.1.2.4 networks

This file is equivalent to the Unix `/etc/networks` file. It contains information regarding the known networks which comprise the Internet. For each network a single line should be present with the following information:

```
official network name
network number
aliases
```

Items are separated by any number of blanks and/or tab characters. A "#" indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. This file is normally created from the official network data base maintained at the Network Information Control Center (NIC), though local changes may be required to bring it up to date regarding unofficial aliases and/or unknown networks.

Network number may be specified in the conventional dotted decimal notation. Network names may contain any printable character other than a field delimiter, newline, or comment character.

7.1.2.5 protocols

This file is equivalent to the Unix `/etc/protocols` file. It contains information regarding the known protocols used in the Internet. For each protocol a single line should be present with the following information:

```
official protocol name
protocol number
aliases
```

Items are separated by any number of blanks and/or tab characters. A "#" indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

Protocol names may contain any printable character other than a field delimiter, newline, or comment character.

7.1.2.6 routes

The primary purpose of this file is to avoid configuring the routing information for the TCP/IP stack through the `AddNetRoute` command. Instead of invoking the `AddNetRoute` command you specify the routing setup in this file instead. This why the options read from this file are identical to the options of the `AddNetRoute` command.

`ppp_dialer` processes the configuration file one line at a time. Each line starts with a keyword (e.g. '`interface`'), which is then followed by the value to assign. A `=` between the keyword and the value is optional.

Caution: If the value you need to assign contains blank spaces, asterisks (`*`) or double quote characters (`"`), you will need to rewrite the value for it to be processed correctly.

- Blank spaces

  The value needs to be enclosed in double quotes. For example, `Example text` would have to be rewritten as `"Example text"`. This is necessary because the processing of the value otherwise stops at the first blank space.

  Once you need to enclose the value in double quotes, you will need to check if you now need to rewrite it further.

- Asterisks

  If the value is enclosed in double quotes, each asterisk has to be replaced by two asterisk characters.

  For example, `"ATD*101"` would have to be rewritten as `"ATD**101"`. This is necessary because the asterisk character within a quoted value has a special meaning.

If the character which directly follows the `*` is one of `e`, `E`, `n`, or `N`, then a control character (Esc or Line Feed, respectively) will be produced. This is likely not what you want.

If the character which directly follows the `*` is `*`, a `*` will be produced. Hence, `**` produces `*`.

If the character which directly follows the `*` is `"`, a `"` will be produced. Hence, `*"` produces `"`.

If any other character follows the `*`, then that character will be ignored. This is likely not what you want.

- Double quotes

  If the value is enclosed in double quotes and the value itself contains double quotes, e.g. when used in a password string, each double quote has to be replaced by `*"` and the result then has to be enclosed in double quotes itself.

  For example, `#;$ "&.` contains blank spaces and also a double quote character which requires that it is rewritten and then enclosed in double quotes as well. Hence, `#;$ "&.` must be changed to `#;$ *"&.` and then enclosed in double quotes: `"#;$ *"&."`

You may want to limit the use of blank spaces, asterisks and double quote characters in your configuration file.

Each line of the `routes` file is read and processed according to the following template:

```
DST=DESTINATION/K,HOSTDST=HOSTDESTINATION/K,NETDST=NETDESTINATION/K,
VIA=GATEWAY/K,DEFAULT=DEFAULTGATEWAY/K
```

Blank lines, or lines which start with the `"#"` character will be ignored.

The individual options have the following meanings:

'`DST=DESTINATION/K`'
: The destination address of a route (or in other words, where the route to be added leads to). This must be an IP address or a symbolic name. Some routes may require you to specify a gateway address through which the route has to pass. Depending upon the address you specify, the protocol stack will attempt to figure out whether the destination is supposed to be a host or a network.

'`HOSTDST=HOSTDESTINATION/K`'
: Same as the '`DST=DESTINATION/K`' parameter, except that the destination is assumed to be a host (rather than a network).

'`NETDST=NETDESTINATION/K`'
: Same as the '`DST=DESTINATION/K`' parameter, except that the destination is assumed to be a network (rather than a host).

'`VIA=GATEWAY/K`'
: This parameter complements the route destination address; it indicates the address to which a message should be sent for it to be passed to the destination. This must be an IP address or a symbolic name.

'`DEFAULT=DEFAULTGATEWAY/K`'
: This parameter selects the default gateway address (which must be specified as an IP address or a symbolic host name) all messages are sent to which don't have any particular other routes associated with them. Another, perhaps less misleading name for *default gateway address* is *default route*.

You must either specify the default route, or specify a route with a destination address. Note that no duplicates are allowed, i.e. there can be only a single default route and the same route may not be established twice.

If you use the 'DEFAULT=DEFAULTGATEWAY/K' option, all other destination addresses you may have specified will be ignored. Only one of 'DESTINATION', 'HOSTDESTINATION' or 'NETDESTINATION' will be used; choose only one.

Note that the routes file is not strictly required for proper operations, and its contents may even cause conflicts. Both the DHCP and PPP/PPPoE configuration processes will pick their own default route, and these may conflict with the settings in the routes file.

Note that Roadshow will track changes to the DEVS:Internet/routes file and automatically reread and process the file if necessary.

7.1.2.7 rpc

This file is equivalent to the Unix /etc/rpc file. It contains user readable names that can be used in place of rpc program numbers. Each line has the following information:

```
name of server for the rpc program
rpc program number
aliases
```

Items are separated by any number of blanks and/or tab characters. A "#" indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

7.1.2.8 servers

The purpose of the DEVS:Internet/servers file is to tell Roadshow which server programs should be started whenever a connection is made to the local computer. The contents of the file describe how the connection must be made in order to start a server program, and how this server program may be started. While the format of the DEVS:Internet/servers file is different, it basically serves the same purpose as the Unix /etc/inetd.conf file as used by the Internet superserver.

ppp_dialer processes the configuration file one line at a time. Each line starts with a keyword (e.g. 'interface'), which is then followed by the value to assign. A = between the keyword and the value is optional.

Caution: If the value you need to assign contains blank spaces, asterisks (*) or double quote characters ("), you will need to rewrite the value for it to be processed correctly.

- Blank spaces

  The value needs to be enclosed in double quotes. For example, Example text would have to be rewritten as "Example text". This is necessary because the processing of the value otherwise stops at the first blank space.

  Once you need to enclose the value in double quotes, you will need to check if you now need to rewrite it further.

- Asterisks

  If the value is enclosed in double quotes, each asterisk has to be replaced by two asterisk characters.

  For example, "ATD*101" would have to be rewritten as "ATD**101". This is necessary because the asterisk character within a quoted value has a special meaning.

  If the character which directly follows the * is one of e, E, n, or N, then a control character (Esc or Line Feed, respectively) will be produced. This is likely not what you want.

  If the character which directly follows the * is *, a * will be produced. Hence, ** produces *.

  If the character which directly follows the * is ", a " will be produced. Hence, *" produces ".

If any other character follows the `*`, then that character will be ignored. This is likely not what you want.

- Double quotes

  If the value is enclosed in double quotes and the value itself contains double quotes, e.g. when used in a password string, each double quote has to be replaced by `*"` and the result then has to be enclosed in double quotes itself.

  For example, `#;$ "&.` contains blank spaces and also a double quote character which requires that it is rewritten and then enclosed in double quotes as well. Hence, `#;$ "&.` must be changed to `#;$ *"&.` and then enclosed in double quotes: `"#;$ *"&."`

You may want to limit the use of blank spaces, asterisks and double quote characters in your configuration file.

Each line of the `servers` file is read and processed according to the following template:

```
NAME/A,INACTIVE/S,STREAM/S,DGRAM=DATAGRAM/S,SEQPACKET=SEQUENCEDPACKET/S,
RAW/S,WAIT/S,DOS/S,NOREQ/S,PATH/K/M,PRI=PRIORITY/K/N,MAXHITS/K/N,
STACK=STACKSIZE/K/N,PROGRAM/F
```

Blank lines, or lines which start with the `"#"` character will be ignored.

If the program name used with the '`NAME`' parameter, or the path name used with the '`PATH`' parameter, respectively, contains blank spaces then they *must* be enclosed in double quotes. Otherwise the entire configuration file line may be ignored or misinterpreted.

The individual parameters and options have the following meanings:

'`NAME/A`' This selects the service name, which must be listed in the file `DEVS:Internet/services`.

'`INACTIVE/S`'
: This option disables the service. If omitted, the service is assumed to be enabled.

'`STREAM/S`'
'`DGRAM=DATAGRAM/S`'
'`SEQPACKET=SEQUENCEDPACKET/S`'
'`RAW/S`' Choose one of these four to select the connection type (also known as *socket type*). Choosing '`STREAM`' will use a TCP connection, and all others will use '`UDP`' connections.

'`WAIT/S`' This option specifies that the server which is invoked will take over the connection associated with the service access point, and that the Internet superserver should wait for the server to exit before listening to a new service request.

Datagram servers must use '`WAIT`', as they are always invoked with the original datagram socket bound to the specified service address. These servers must read at least one datagram from the socket before exiting.

'`DOS/S`'

'`PATH/K/M`'
: The '`DOS`' option specifies that the server to invoke will be launched with its standard input/output streams mapped to the connection associated with the service access point. This makes it trivial to have plain shell programs act as Internet servers.

  Typically, the Internet superserver will only look into the `C:` drawer when looking for the program to start. To specify which directories to search for commands, use the '`PATH`' option. Multiple path names can be specified, e.g. '`Path SYS:RexxC SYS:Tools`'.

  Note that you do not need to use the '`PATH`' option if you know exactly where the program is found that should be started. For example, the options `PATH Samba:bin`

`PROGRAM smbd` have exactly the same effect as using `PROGRAM Samba:bin/smbd` would have.

The '`PATH`' option is useful if there is more than one drawer in which the program to be started could be found. In all other cases it should be sufficient to just add the path name to the program name given with the '`PROGRAM`' parameter.

Caution: If the path name contains blank spaces then you *must* enclose it in double quotes (e.g. `PATH "volume:path name/file name"`) or the entire configuration file line may be ignored or misinterpreted!

'`NOREQ/S`' If a command or script is invoked as part of a service which may access local volumes or files, it can end up trying to open a file or drawer on a volume which is not currently mounted. You may see the familiar "Please insert disk" requester window open which requires you to take action.

This is problematic if you want your Amiga to launch commands and scripts without requiring you to be present at all times. The '`NOREQ`' option, which is short for "no requester windows please", will start these commands or scripts in a manner which prevents the requester windows from appearing. This option is useful for services which use the '`DOS`' option and for the built-in remote shell server function.

Please note that if you invoke commands or scripts which open windows on the screen of your Amiga on their own accord, rather than as a by-product of accessing files or drawers, then the '`NOREQ`' option will not prevent these windows from opening. Also, if such commands or scripts start further commands or scripts, then the effect of the '`NOREQ`' option may not transfer to these.

'`PRI=PRIORITY/K/N`'

This option selects the Task priority for the server program to be started. If omitted, a priority of 0 will be used.

'`MAXHITS/K/N`'

This option can be used to change how many times per minute this server may be invoked. The default is to permit a server to be invoked up to 1500 times per minute. The server is temporarily disabled when this limit is reached, and another minute may have to pass before it will be re-enabled.

'`STACK=STACKSIZE/K/N`'

This selects the stack size which the server should be able to use. If omitted, the system default stack size will be used.

'`PROGRAM/F`'

This is where you specify the name of the program, and all its parameter, which will be used when this server is started. Note that the name '`internal`' has special meaning, and will select a built-in server for the following services: echo, discard, time, daytime and chargen.

Caution: If the program name contains blank spaces then you *must* enclose it in double quotes (e.g. `PROGRAM "volume:path name/program name"`) or the entire configuration file line may be ignored or misinterpreted!

Note that Roadshow will track changes to the `DEVS:Internet/servers` file and automatically reread and process the file if necessary.

7.1.2.9 How to convert inetd.conf settings for use with DEVS:Internet/servers

The purpose of the `DEVS:Internet/servers` file is the same as the Unix `/etc/inetd.conf` file, as used by the Internet superserver. But the format of the file is not the same, since some of the features of a Unix system cannot be reproduced on the Amiga.

Here is how the format of each line of the `/etc/inetd.conf` file looks like (optional parameters are in square brackets, and the '`|`' character means that you have to pick one of the choices on its left or right):

```
service name
socket type
protocol
wait[.maximum number of connections minute]|nowait[.maximum number of connections minute]
user[:group]
path to server program
server program name and arguments
```

To illustrate, here is an example of how the Samba server programs `smbd`, `nmbd` and `swat` may be used in an `/etc/inetd.conf` file on a Unix system:

```
netbios-ssn stream tcp nowait root /usr/sbin/smbd smbd
netbios-ns dgram udp wait root /usr/sbin/nmbd nmbd
swat stream tcp nowait.400 root  /usr/sbin/swat swat -a
```

From the example above, the configuration of the `smbd` server breaks down into the following parameters:

'`netbios-ssn`'
: This is the '`service name`' parameter

'`stream`'
: This is the '`socket type`' parameter

'`tcp`'
: This is the '`protocol`' parameter

'`nowait`'
: This is the wait/nowait option. In this case, the '`nowait`' option was selected for this server; no limit on the number of connections per minute is specified

'`root`'
: This is the '`user`' parameter; no group name is specified

'`/usr/sbin/smbd`'
: This is the '`path to server program`' parameter

'`smbd`'
: This is the '`server program name and arguments`' parameter

In order to convert the `/etc/inetd.conf` style parameters into the format which the `DEVS:Internet/servers` file uses, you need to know how the respective parameters match up. This is a list of `/etc/inetd.conf` parameters with their respective counterparts in the `DEVS:Internet/servers` file:

'`service name`'
: This corresponds to the '`NAME/A`' parameter. In the example for '`smbd`' above, the '`netbios-ssn`' parameter would be written as '`NAME=netbios-ssn`' instead.

'`socket type`'
: This corresponds to one of the '`STREAM/S`', '`DGRAM=DATAGRAM/S`', '`SEQPACKET=SEQUENCEDPACKET/S`' or '`RAW/S`' parameters. In the example for '`smbd`' above, the '`stream`' parameter would be written as '`STREAM`' instead.

'`protocol`'
: There is no corresponding parameter for this item. In the example for '`smbd`' above, the '`tcp`' parameter would be omitted.

'`wait/nowait option`'
: There is no corresponding parameter for this item, or rather, there is only a counterpart for '`wait`' but not for '`nowait`'. In the example for '`smbd`' above, the '`nowait`' parameter would be omitted.

'`user`'
: There is no corresponding parameter for this item. In the example for '`smbd`' above, the '`root`' parameter would be omitted.

'path to server program'
: There is no corresponding parameter for this item. In the example for 'smbd' above, the '/usr/sbin/smbd' parameter would be omitted.

'server program name and arguments'
: This corresponds to 'PROGRAM/F' parameter. In the example for 'smbd' above, the 'smbd' parameter would be written as 'PROGRAM=smbd' instead.

To give you an idea how the Samba server example /etc/inetd.conf file could be converted for Roadshow, here is how the same lines might look like in the DEVS:Internet/servers file:

```
NAME=netbios-ssn stream PROGRAM=Samba:bin/smbd
NAME=netbios-ns datagram WAIT PROGRAM=Samba:bin/nmbd
NAME=swat stream MAXHITS=400 PROGRAM=Samba:bin/swat -a
```

7.1.2.10 services

This file is equivalent to the Unix /etc/services file. It contains information regarding the known services available in the Internet. For each service a single line should be present with the following information:

```
official service name
port number
protocol name
aliases
```

Items are separated by any number of blanks and/or tab characters. The port number and protocol name are considered a single item; a "/" is used to separate the port and protocol (e.g. '512/tcp'). A "#" indicates the beginning of a comment; subsequent characters up to the end of the line are not interpreted by the routines which search the file.

Service names may contain any printable character other than a field delimiter, newline, or comment character.

7.1.2.11 users

This purpose of this file is similar to the Unix /etc/passwd file, but uses a different format to store its information.

The contents of this file are used by software which needs to translate the names of user, and which groups the respective user is a member of, into numbers. A typical application would be the use of the Amiga NFS (network file system) client.

ppp_dialer processes the configuration file one line at a time. Each line starts with a keyword (e.g. 'interface'), which is then followed by the value to assign. A = between the keyword and the value is optional.

Caution: If the value you need to assign contains blank spaces, asterisks (*) or double quote characters ("), you will need to rewrite the value for it to be processed correctly.

- Blank spaces

  The value needs to be enclosed in double quotes. For example, Example text would have to be rewritten as "Example text". This is necessary because the processing of the value otherwise stops at the first blank space.

  Once you need to enclose the value in double quotes, you will need to check if you now need to rewrite it further.

- Asterisks

  If the value is enclosed in double quotes, each asterisk has to be replaced by two asterisk characters.

  For example, "ATD*101" would have to be rewritten as "ATD**101". This is necessary because the asterisk character within a quoted value has a special meaning.

If the character which directly follows the * is one of e, E, n, or N, then a control character (Esc or Line Feed, respectively) will be produced. This is likely not what you want.

If the character which directly follows the * is *, a * will be produced. Hence, ** produces *.

If the character which directly follows the * is ", a " will be produced. Hence, *" produces ".

If any other character follows the *, then that character will be ignored. This is likely not what you want.

- Double quotes

  If the value is enclosed in double quotes and the value itself contains double quotes, e.g. when used in a password string, each double quote has to be replaced by *" and the result then has to be enclosed in double quotes itself.

  For example, #;$ "&. contains blank spaces and also a double quote character which requires that it is rewritten and then enclosed in double quotes as well. Hence, #;$ "&. must be changed to #;$ *"&. and then enclosed in double quotes: "#;$ *"&."

You may want to limit the use of blank spaces, asterisks and double quote characters in your configuration file.

Each line of the users file is read and processed according to the following template:

```
NAME/A,PASSWORD/K,UID/A/N,GID/A/N,GECOS,DIR,SHELL
```

Blank lines, or lines which start with the "#" character will be ignored.

The individual options have the following meanings:

'NAME/A' The name of the user (this is often referred to as the *login name*).

'PASSWORD/K'
: The password of the user, stored as plain text.

'UID/A/N' The numeric ID value for this user.

'GID/A/N' The numeric ID value for the so-called *login group* which this user is a member of. Additional groups which this user is a member of may be configured through the groups configuration file.

'GECOS' General information about the user, often used to store the full name of the user.

'DIR' The home directory of this user.

'SHELL' The name of the login shell of this user.

Note that the users and groups files are considered obsolete and are provided only in order to assist the few applications which require them. It is recommended that you do not use the users file to store sensitive information, such as login and password data.

7.1.3 PPP/PPPoE configuration files in "S:PPP-Configurations"

When connecting to a PPP or PPPoE service, the command which makes the connection (either ppp_dialer for PPP or ppp_connector for PPPoE) needs a set of instructions which detail how the connection should be set up.

These instructions govern both the internal workings of the PPP protocol, and options specific to the connection medium. For example, a PPP connection can be made through a modem or nullmodem cable, and for the modem connection to be made, initialization and dialing commands will have to be sent to the modem.

Each PPP/PPPoE configuration file consists of a number of lines of text, each of which can contain a comment or a configuration option.

Empty lines ignored when the respective text file is being processed. A line which begins with the "#" character is considered a comment and will be ignored, too.

A configuration option always starts with a keyword, such as `device`, which is followed by a "=" character, which is followed by the option value. There can be any number of blank spaces preceding or following each of these.

7.1.3.1 ppp-ethernet.device

This section describes the contents of the configuration files used by the `ppp_connector` program.

`ppp_dialer` processes the configuration file one line at a time. Each line starts with a keyword (e.g. '`interface`'), which is then followed by the value to assign. A `=` between the keyword and the value is optional.

Caution: If the value you need to assign contains blank spaces, asterisks (`*`) or double quote characters (`"`), you will need to rewrite the value for it to be processed correctly.

- Blank spaces

  The value needs to be enclosed in double quotes. For example, `Example text` would have to be rewritten as `"Example text"`. This is necessary because the processing of the value otherwise stops at the first blank space.

  Once you need to enclose the value in double quotes, you will need to check if you now need to rewrite it further.

- Asterisks

  If the value is enclosed in double quotes, each asterisk has to be replaced by two asterisk characters.

  For example, `"ATD*101"` would have to be rewritten as `"ATD**101"`. This is necessary because the asterisk character within a quoted value has a special meaning.

  If the character which directly follows the `*` is one of `e`, `E`, `n`, or `N`, then a control character (Esc or Line Feed, respectively) will be produced. This is likely not what you want.

  If the character which directly follows the `*` is `*`, a `*` will be produced. Hence, `**` produces `*`.

  If the character which directly follows the `*` is `"`, a `"` will be produced. Hence, `*"` produces `"`.

  If any other character follows the `*`, then that character will be ignored. This is likely not what you want.

- Double quotes

  If the value is enclosed in double quotes and the value itself contains double quotes, e.g. when used in a password string, each double quote has to be replaced by `*"` and the result then has to be enclosed in double quotes itself.

  For example, `#;$ "&.` contains blank spaces and also a double quote character which requires that it is rewritten and then enclosed in double quotes as well. Hence, `#;$ "&.` must be changed to `#;$ *"&.` and then enclosed in double quotes: `"#;$ *"&."`

You may want to limit the use of blank spaces, asterisks and double quote characters in your configuration file.

Each configuration file option is parsed according to the following template:

```
INTERFACE/K,DEVICE/K,UNIT/K/N,RAW/K,BYPASS/K,READPACKETS/K/N,
WRITEPACKETS/K/N,SERVICE/K,AC=ACCESSCONCENTRATOR/K,LOCALADDRESS=LOCALIP/K,
REMOTEADDRESS=REMOTEIP/K,DNS1ADDRESS=DNS1IP/K,DNS2ADDRESS=DNS2IP/K,
MAXFAIL/K/N,MAXTERM/K/N,MAXCONFIG/K/N,TIMEOUT/K/N,MAXRECONFIGURE/K/N,
MTU/K/N,IDLETIMEOUT/K/N,PEERIDLETIMEOUT/K/N,SETENV/K,SENDID/K,REJECTPAP/K,
PAPTIMEOUT/K/N,PAPRETRY/K/N,DUMMYREMOTEADDRESS/K,CONNECTTIMEOUT/K/N,
LOGIN/K,PASSWORD/K,LOGFILE/K,LOG/K
```

Here is what the individual parameters do:

'INTERFACE/K'
: This parameter is mandatory. It must refer to the networking interface you configured for `bsdsocket.library`, e.g. `PPPoE`. The networking driver attached to the interface must be `ppp-ethernet.device`, or things won't work.

'DEVICE/K'
: You must supply the name of the device driver which should provide the PPP link layer, e.g. `a2065.device` or `ariadne.device`. You can either supply a complete path name (e.g. `DEVS:Networks/a2065.device`) or just specify the device name (e.g. `a2065.device`) in which case `ppp-ethernet.device` will look into `DEVS:Networks` all by itself.

  Note: You cannot specify just any network device driver. It must be an Ethernet hardware driver.

'UNIT/K/N'
: This parameter is optional. If you have multiple cards of the same type installed, each device unit corresponds to another board. Make sure you pick the right one.

  Default for this option is '`0`'.

'LOCALADDRESS/K'
: Instead of allowing the peer to assign an IP address to the local host, a static IP address can be specified as the default. Whether the peer will accept this choice is up to the following PPP negotiation process.

  The IP address must be specified in dotted decimal notation, e.g. '`1.2.3.4`'.

  Default is to let the PPP negotiation process choose an IP address for the local host.

'REMOTEADDRESS/K'
: Instead of waiting for the peer to report its IP address as part of the PPP negotiation process, a default address can be used in its place. If such a default address is used, whatever the peer will report as its own IP address will be ignored.

  The IP address must be specified in dotted decimal notation, e.g. '`1.2.3.4`'.

  Default is to wait for the peer to report its own IP address.

'DNS1ADDRESS/K'
'DNS2ADDRESS/K'
: Instead of asking the peer to report the addresses of the domain name servers, defaults can be used instead. Whether the peer will accept this choice is up to the following PPP negotiation process.

  The IP addresses must be specified in dotted decimal notation, e.g. '`1.2.3.4`'.

  Default is to let the PPP negotiation process report the domain name server addresses.

'MAXFAIL/K/N'
: This option controls a PPP protocol parameter, the maximum number of negative configuration acknowledgements to be sent before it switches to reject those options. You won't need to change this.

  Default for this option is '`5`'.

'MAXTERM/K/N'
: This option controls a PPP protocol parameter, the maximum number of termination requests to be sent before the respective network or link protocol gives up. You won't need to change this.

  Default for this option is '`2`'.

'MAXCONFIG/K/N'
: This option controls a PPP protocol parameter, the maximum number of configuration requests to be sent before the respective network or link protocol gives up. You won't need to change this.

  Default for this option is '10'.

'TIMEOUT/K/N'
: This option controls a PPP protocol parameter, the number of seconds that have to pass before the respective network or link protocol will retry to do whatever didn't work during the last attempt. You won't need to change this.

  Read carefully: This is a PPP configuration parameter which affects the behaviour of the protocol itself. It has nothing, I repeat, *nothing*, to do with the connection establishment process!

  Default for this option is '3' seconds.

'MAXRECONFIGURE/K/N'
: The PPP protocol negotiates its operating parameters with the remote. Due to misconfigurations or implementation errors, this process may end up in an infinite loop and never come to conclusion. This is called *convergence failure*. To catch such situations, this implementation maintains a counter which clocks each negotiation reconfiguration attempt. If more attempts are made than are allowed, then the process will be aborted.

  Default for this option is '20'.

'MTU/K/N' This controls the maximum number of characters the driver will send to the remote in each packet. You shouldn't really need to change this, but they say that smaller MTUs can give better responsiveness at the expense of performance. Note that the MTU cannot be larger than 1492 bytes because each PPP frame, including PPPoE fluff, must fit completely into an Ethernet frame of 1500 bytes.

  Default for this option is '1492' bytes.

'IDLETIMEOUT/K/N'
: The driver can monitor its I/O streams and decide when and for how long nothing useful was being done. You can tell the driver to close the connection if it has been idle for too long by specifying the number of seconds of idleness after which the session is to be closed.

  Default for this option is '0' seconds which disables the timeout.

'PEERIDLETIMEOUT/K/N'
: If the peer did not send any responses during the previous 10 seconds, the driver will send an echo request to check whether the link is still open. If the peer does not respond to this request for a certain period of time, then the link will be assumed to be dead and the session will be closed. That period of time is configurable and must be a multiple of 10 seconds.

  Default for this option is '60' seconds.

'SETENV/K'
: For those applications which need it, the driver can store the information it obtains upon connection in environment variables. The names of the variables are set according to the name and unit number of the driver used. For example, for `ppp-ethernet.device` unit 0, the following variables would be set:

  'ppp-ethernet0.local_address'
  : The IP address assigned to the local PPP client.

'ppp-ethernet0.peer_address'
The IP address used by the peer (the server).

'ppp-ethernet0.dns1_address'
The IP address of the primary domain name server.

'ppp-ethernet0.dns2_address'
The IP address of the secondary domain name server.

These environment variables will be set once the driver has gone online. They will be removed once it has gone offline again.

Default for this option is 'SETENV=OFF', i.e. these environment variables will not be set.

'SENDID/K'
When starting up, and during significant state transitions, the driver will issue packets which identify itself to the peer. This is not always welcome, and some peers may fail to react properly to them. The default action taken, however, should be safe. If you suspect that the identification packets are not welcome, you should disable them.

Default for this option is 'SENDID=ON'.

'REJECTPAP/K'
Some PPPoE servers may request that the login and password are transmitted unencrypted in the open. This presents a huge security problem since PPPoE transmits all its frames over a broadcast medium which can be monitored using packet sniffers. An attack on your account would be trivial to execute. To tell the PPPoE server not to use a plain text authentication protocol (*Password Authentication Protocol* also known as *PAP*), but rather to resort to the much safer *Challenge Handshake Authentication Protocol* (also known as *CHAP*) use the 'REJECTPAP=ON' option. Note that if the server is stubborn, it might refuse to 'upgrade' its authentication process. In that case, the authentication is likely to fail and you may be required to disable the 'REJECTPAP' option.

Default for this option is 'REJECTPAP=OFF'.

'PAPTIMEOUT/K/N'
The Password Authentication Protocol requires that the client takes the initiative and asks the server to get it online. These authentication requests are resent periodically, with a small delay between the individual requests.

Default for this option is '3' seconds.

'PAPRETRY/K/N'
The Password Authentication Protocol will try several times to get the server to listen and finally get it online. This process does not run forever. At some point of time it has to stop, so that the connection can be closed. This option defines how many times the request is resent.

Default for this option is '10' times.

'DUMMYREMOTEADDRESS/K'
During the negotiation process which provides the local host with the IP address the server assigned to it, the server should also come forward and reveal the IP address to which it responds. While this is not strictly necessary, it is extremely helpful to know the peer's IP address. Unfortunately, some servers will keep this information to themselves and refuse to provide any information at all. You can do two things to resolve this problem. First, you can specify a default remote address with the

'REMOTEADDRESS' option (the address '192.168.1.1' is recommended). Take care: the address you pick must not be used by any other interface that is currently known to the TCP/IP stack. Second, you can have the PPP driver pick a dummy IP address at random. This is what the 'DUMMYREMOTEADDRESS' option controls.

Default for this option is 'ON'.

'LOGIN/K'

'PASSWORD/K'

This is where you provide the authentication information that may be required by the PPPoE server you want to connect to.

Note that PPP does not normally require authentication, which is why these two options are by default empty.

'LOGFILE/K'

This selects the name of the log file where progress reports and error messages will be stored for future reference. The file will be created if it does not yet exist, and if it exists, new data will be appended to it. Note that no log entries will be written unless you specify log options with the 'LOG' parameter.

Default for this option is an empty name, i.e. no log file will be created.

'LOG/K'

This option selects the events and actions which log entries should be written for. This must be one of the following, or a combination, separated by commas and/or blanks:

'LINK' Log link layer events.

'CONNECT' Log connection establishment events.

'LCP' Log Link Control Protocol events.

'AUTH' Log authentication events.

'IPCP' Log IP Control Protocol events.

'LQR' Log link quality reports, both incoming and outgoing.

'FRAMESIN'
Log the contents of incoming frames. **Note:** this can generate a lot of logging information!

'FRAMESOUT'
Log the contents of outgoing frames. **Note:** this can generate a lot of logging information!

'ALL' Log all of the above. **Note:** this can generate a lot of debugging information!

Note that for log entries to be written, you have to specify a log file to write them to using the 'LOGFILE' parameter. To be on the safe side, create the log file in nonvolatile memory, e.g. RAD:, rather than on a hard disk drive.

The following parameters are specific to PPPoE:

'SERVICE/K/N'

According to the PPPoE standard, servers may offer more than one service, and it might be important to select the service you want. This option is responsible for picking the right one. What the service name is supposed to be is something your

ISP should have told you. If you don't know what it means, you'd better ignore this option and accept the default.

Default for this option is an empty string, which will end up choosing the first service advertized as being available.

'AC=ACCESSCONCENTRATOR/K/N'

PPPoE sessions, through which the network data is exchanged, are established by the local client software and the remote server, which is known as the *access concentrator*. By default, the driver will pick the first access concentrator that responds to the request to open a new session, but you can request that only a particular access concentrator should be used.

Default for this option is an empty string, which will end up choosing the first access concentrator that responds to the session request sent by the local host.

'CONNECTTIMEOUT/K/N'

Making the connection to the PPPoE server should be instantaneous, but to play things safe, we're going to wait a little for the remote to respond and a session to be opened. If we didn't get a confirmation for a session to be opened, the connection attempt will be aborted.

Default for the session open timeout is 30 seconds.

'BYPASS/K'

The PPPoE protocol wraps PPP frames into Ethernet frames, thereby adding its own state information on every frame that is sent. When running, PPPoE really does only two things: it picks up incoming Ethernet frames, strips the PPPoE header and hands over the IP packet contained inside to the TCP/IP stack; or it receives an IP packet, adds a PPPoE header and transmits it in an Ethernet frame.

As you might imagine, there is a lot of data copying involved to hammer the 'payload', in this case the IP packet, into the desired shape. But this need not be the case. You don't really have to copy the IP packet back and forth with every transition. You can bypass the temporary buffer the protocol uses.

Bypassing this intermediate copying stage can save time and bandwidth, thereby increasing overall data throughput. However, there is a catch in that some things don't work so well if you're bypassing the temporary buffer: DMA access will be disabled, and so will the word aligned copy option. This is because the construction of the PPPoE frame may be unable to guarantee that the data copy process is compatible with the alignment requirements of the DMA and word aligned access methods. Another catch is in that the copying process may have to temporarily disable interrupt processing to do its job. This may be costly, but then maybe not. See for yourself: you might actually see the transmission speed increase noticeably.

Default for this option is 'BYPASS=OFF'.

'READPACKETS/K/N'
'WRITEPACKETS/K/N'

The `ppp-ethernet.device` has to queue multiple read and write packets, as otherwise incoming network data may be lost or outgoing data might not find its way to the wire. The number of packets that can be queued is a configuration option which both affects reliability and speed. Choose too little, and you'll lose data. Choose more and – up to a point – performance may improve. Note that for each queued packet, a bit more than 1500 bytes of memory will be allocated.

Defaults for these options are 'READPACKETS=16' and 'WRITEPACKETS=16'.

'RAW/K'

Some Ethernet card drivers have a problem in that they do not transmit PPPoE frames correctly, but rather misinterpret them as IEEE 802.3 framing information. For such drivers it is recommended to have `ppp-ethernet.device` construct complete Ethernet link level headers for the packets to be sent. This requires that so-called 'raw' frames are sent, which can be activated with the '`RAW=ON`' switch.

So far only one Ethernet card driver is known to require this switch: it is `ariadne.device` prior to version 1.51.

Default for this option is '`RAW=OFF`'.

7.1.3.2 ppp-serial.device

This section describes the contents of the configuration files used by the `ppp_dialer` program.

`ppp_dialer` processes the configuration file one line at a time. Each line starts with a keyword (e.g. '`interface`'), which is then followed by the value to assign. A `=` between the keyword and the value is optional.

Caution: If the value you need to assign contains blank spaces, asterisks (`*`) or double quote characters (`"`), you will need to rewrite the value for it to be processed correctly.

- Blank spaces

  The value needs to be enclosed in double quotes. For example, `Example text` would have to be rewritten as `"Example text"`. This is necessary because the processing of the value otherwise stops at the first blank space.

  Once you need to enclose the value in double quotes, you will need to check if you now need to rewrite it further.

- Asterisks

  If the value is enclosed in double quotes, each asterisk has to be replaced by two asterisk characters.

  For example, `"ATD*101"` would have to be rewritten as `"ATD**101"`. This is necessary because the asterisk character within a quoted value has a special meaning.

  If the character which directly follows the `*` is one of `e`, `E`, `n`, or `N`, then a control character (Esc or Line Feed, respectively) will be produced. This is likely not what you want.

  If the character which directly follows the `*` is `*`, a `*` will be produced. Hence, `**` produces `*`.

  If the character which directly follows the `*` is `"`, a `"` will be produced. Hence, `*"` produces `"`.

  If any other character follows the `*`, then that character will be ignored. This is likely not what you want.

- Double quotes

  If the value is enclosed in double quotes and the value itself contains double quotes, e.g. when used in a password string, each double quote has to be replaced by `*"` and the result then has to be enclosed in double quotes itself.

  For example, `#;$ "&.` contains blank spaces and also a double quote character which requires that it is rewritten and then enclosed in double quotes as well. Hence, `#;$ "&.` must be changed to `#;$ *"&.` and then enclosed in double quotes: `"#;$ *"&."`

You may want to limit the use of blank spaces, asterisks and double quote characters in your configuration file.

Each option is parsed according to the following template:

```
INTERFACE/K,DEVICE/K,UNIT/K/N,BPS=SPEED/K/N,WRITEREQUESTS/K/N,READREQUESTS/K/N,
BUFFERSIZE/K/N,CD=CHECKCARRIER/K,7WIRE=RTSCTS/K,SHARED/K,NULLMODEM/K,EOF/K,
LOCALADDRESS=LOCALIP/K,REMOTEADDRESS=REMOTEIP/K,DNS1ADDRESS=DNS1IP/K,
```

```
DNS2ADDRESS=DNS2IP/K,MAXFAIL/K/N,MAXTERM/K/N,MAXCONFIG/K/N,TIMEOUT/K/N,
MAXRECONFIGURE/K/N,MTU/K/N,IDLETIMEOUT/K/N,PEERIDLETIMEOUT/K/N,ACCM/K,PFC/K,
AACFC/K,VJHC/K,IGNOREFCS/K,SETENV/K,SENDID/K,REJECTPAP/K,PAPTIMEOUT/K/N,
PAPRETRY/K/N,DUMMYREMOTEADDRESS/K,INIT/K,DIAL/K,DIALTIMEOUT/K/N,LOGIN/K,
PASSWORD/K,HANGUP/K,LOGFILE/K,LOG/K
```

Here is what the individual template parameters do:

'`INTERFACE/K`'
: This parameter is mandatory. It must refer to the networking interface you configured for `bsdsocket.library`, e.g. `PPP`. The networking driver attached to the interface must be `ppp-serial.device`, or things won't work.

'`DEVICE/K`'
: You must supply the name of the device driver which should provide the PPP link layer, e.g. `serial.device` or `duart.device`. Note that the use of `serial.device` or its various replacements, such as `baudbandit.device` is strongly discouraged since, no matter what, the hardware's flaws cannot be cured in software. If you insist on using the Amiga's built-in serial port you're on your own.

'`UNIT/K/N`'
: This parameter is optional. For multiserial boards each unit number corresponds to another port. Pick the right port or omit this option.

  Default for this option is '`0`'.

'`WRITEREQUESTS/K/N`'
: The driver uses asynchronous I/O when sending data to the peer. This is intended to improve performance during the protocol negotiation stage. More than one I/O request is involved in this procedure, and the more are available, the better the throughput could become.

  Default for this option is '`8`'.

'`LOCALADDRESS/K`'
: Instead of allowing the peer to assign an IP address to the local host, a static IP address can be specified as the default. Whether the peer will accept this choice is up to the following PPP negotiation process.

  The IP address must be specified in dotted decimal notation, e.g. '`1.2.3.4`'.

  Default is to let the PPP negotiation process choose an IP address for the local host.

'`REMOTEADDRESS/K`'
: Instead of waiting for the peer to report its IP address as part of the PPP negotiation process, a default address can be used in its place. If such a default address is used, whatever the peer will report as its own IP address will be ignored.

  The IP address must be specified in dotted decimal notation, e.g. '`1.2.3.4`'.

  Default is to wait for the peer to report its own IP address.

'`DNS1ADDRESS/K`'
'`DNS2ADDRESS/K`'
: Instead of asking the peer to report the addresses of the domain name servers, defaults can be used instead. Whether the peer will accept this choice is up to the following PPP negotiation process.

  The IP addresses must be specified in dotted decimal notation, e.g. '`1.2.3.4`'.

  Default is to let the PPP negotiation process report the domain name server addresses.

'`MAXFAIL/K/N`'
: This option controls a PPP protocol parameter, the maximum number of negative configuration acknowledgements to be sent before it switches to reject those options. You won't need to change this.

Default for this option is '5'.

'MAXTERM/K/N'
: This option controls a PPP protocol parameter, the maximum number of termination requests to be sent before the respective network or link protocol gives up. You won't need to change this.

  Default for this option is '2'.

'MAXCONFIG/K/N'
: This option controls a PPP protocol parameter, the maximum number of configuration requests to be sent before the respective network or link protocol gives up. You won't need to change this.

  Default for this option is '10'.

'TIMEOUT/K/N'
: This option controls a PPP protocol parameter, the number of seconds that have to pass before the respective network or link protocol will retry to do whatever didn't work during the last attempt. You won't need to change this.

  Read carefully: This is a PPP configuration parameter which affects the behaviour of the protocol itself. It has nothing, I repeat, *nothing*, to do with the dialer!

  Default for this option is '3' seconds.

'MAXRECONFIGURE/K/N'
: The PPP protocol negotiates its operating parameters with the remote. Due to misconfigurations or implementation errors, this process may end up in an infinite loop and never come to conclusion. This is called *convergence failure*. To catch such situations, this implementation maintains a counter which clocks each negotiation reconfiguration attempt. If more attempts are made than are allowed, then the process will be aborted.

  Default for this option is '20'.

'MTU/K/N'
: This controls the maximum number of characters the driver will send to the remote in each packet. You shouldn't really need to change this, but they say that smaller MTUs can give better responsiveness at the expense of performance.

  Default for this option is '1500' bytes.

'IDLETIMEOUT/K/N'
: The driver can monitor its I/O streams and decide when and for how long nothing useful was being done. You can tell the driver to close the connection if it has been idle for too long by specifying the number of seconds of idleness after which the line is to be closed.

  Default for this option is '0' seconds, which disables the timeout.

'PEERIDLETIMEOUT/K/N'
: If the peer did not send any responses during the previous 10 seconds, the driver will send an echo request to check whether the link is still open. If the peer does not respond to this request for a certain period of time, then the link will be assumed to be dead and the session will be closed. That period of time is configurable and must be a multiple of 10 seconds.

  Default for this option is '60' seconds.

'SETENV/K'
: For those applications which need it, the driver can store the information it obtains upon connection in environment variables. The names of the variables are set

according to the name and unit number of the driver used. For example, for `ppp-serial.device` unit 0, the following variables would be set:

'`ppp-serial0.local_address`'
: The IP address assigned to the local PPP client.

'`ppp-serial0.peer_address`'
: The IP address used by the peer (the server).

'`ppp-serial0.dns1_address`'
: The IP address of the primary domain name server.

'`ppp-serial0.dns2_address`'
: The IP address of the secondary domain name server.

These environment variables will be set once the driver has gone online. They will be removed once it has gone offline again.

Default for this option is '`SETENV=OFF`', i.e. these environment variables will not be set.

'`SENDID/K`'
: When starting up, and during significant state transitions, the driver will issue packets which identify itself to the peer. This is not always welcome, and some peers may fail to react properly to them. The default action taken, however, should be safe. If you suspect that the identification packets are not welcome, you should disable them.

  Default for this option is '`SENDID=ON`'.

'`REJECTPAP/K`'
: Some PPP servers may request that the login and password are transmitted unencrypted in the open. To tell the PPP server not to use a plain text authentication protocol (*Password Authentication Protocol*, also known as *PAP*), but rather to resort to the much safer *Challenge Handshake Authentication Protocol* (also known as *CHAP*) use the '`REJECTPAP=ON`' option.

  Default for this option is '`REJECTPAP=OFF`'.

'`PAPTIMEOUT/K/N`'
: The Password Authentication Protocol requires that the client takes the initiative and asks the server to get it online. These authentication requests are resent periodically, with a small delay between the individual requests.

  Default for this option is '`3`' seconds.

'`PAPRETRY/K/N`'
: The Password Authentication Protocol will try several times to get the server to listen and finally get it online. This process does not run forever. At some point of time it has to stop, so that the connection can be closed. This option defines how many times the request is resent.

  Default for this option is '`10`' times.

'`DUMMYREMOTEADDRESS/K`'
: During the negotiation process which provides the local host with the IP address the server assigned to it, the server should also come forward and reveal the IP address to which it responds. While this is not strictly necessary, it is extremely helpful to know the peer's IP address. Unfortunately, some servers will keep this information to themselves and refuse to provide any information at all. You can do two things to resolve this problem. First, you can specify a default remote address with the

'REMOTEADDRESS' option (the address '192.168.1.1' is recommended). **Take care:** the address you pick must not be used by any other interface that is currently known to the TCP/IP stack. Second, you can have the PPP driver pick a dummy IP address at random. This is what the 'DUMMYREMOTEADDRESS' option controls.

Default for this option is 'ON'.

'LOGIN/K'

'PASSWORD/K'

This is where you provide the authentication information that may be required by the PPP server you want to connect to.

PPP does not normally require authentication, which is why these two options are by default empty.

'LOGFILE/K'

This selects the name of the log file where progress reports and error messages will be stored for future reference. The file will be created if it does not yet exist, and if it exists, new data will be appended to it. Note that no log entries will be written unless you specify log options with the 'LOG' parameter.

Default for this option is an empty name, i.e. no log file will be created.

'LOG/K'

This option selects the events and actions which log entries should be written for. This must be one of the following, or a combination, separated by commas and/or blanks:

'LINK' Log link layer events.

'LCP' Log Link Control Protocol events.

'AUTH' Log authentication events.

'IPCP' Log IP Control Protocol events.

'LQR' Log link quality reports, both incoming and outgoing.

'FRAMESIN'
Log the contents of incoming frames. **Note:** this can generate a lot of logging information!

'FRAMESOUT'
Log the contents of outgoing frames. **Note:** this can generate a lot of logging information!

'DATAIN' Log the contents of the incoming data buffer. **Note:** this can generate a lot of logging information!

'ALL' Log all of the above. **Note:** this can generate a lot of debugging information!

Note that for log entries to be written, you have to specify a log file to write them to using the 'LOGFILE' parameter. To be on the safe side, create the log file in nonvolatile memory, e.g. RAD:, rather than on a hard disk drive.

The following options are specific to PPP:

'7WIRE=RTSCTS/K'

To avoid dropping data due to buffer overruns, you can enable 7 wire handshaking with your modem. This is precisely what the 'RTSCTS=ON' option does. Beware of broken modem cables which don't carry the signalling lines, as the driver may hang

when it enables this handshaking mode. This option is ignored if you are using the 'NULLMODEM=ON' option.

Default for this option is 'RTSCTS=ON'.

'AACFC/K'

This controls a PPP feature called Address-and-Control-Field-Compression, which is a fancy name for a technique that can save two bytes on every IP packet transmitted. If you don't want to save these two bytes, use 'AACFC=OFF'.

Default for this option is 'AACFC=ON'.

'ACCM/K'

This controls a PPP feature by the name of Asynchronous-Control-Character-Map, and allows the protocol to avoid having its frames mangled in the presence of networking equipment that feels about interpreting characters in the range 0..31. You shouldn't normally need to change this, but if you must, then the parameter must be a bit map in which each bit corresponds to a control character to be escaped. For example 'ACCM=%10010000000000' would cause the characters 10 (line feed) and 13 (carriage return) to be escaped before they are committed to the wire. Note that this escaping mechanism will introduce extra overhead and eat speed for every character that needs to be translated prior to transmission. You can also supply the map in hexadecimal notation by omitting the % or by specifying $ or 0x in its place.

Default for this option is '0', i.e. no escaping takes place.

'BPS=SPEED/A/N'

You must supply the transmission speed which the link level device driver will communicate with its peer (or the modem). Popular speeds are beyond 115200 bps, which again discourages the use of the Amiga built-in serial port.

'BUFFERSIZE/K/N'

This parameter is optional. You can specify the size of the serial read buffer here, or accept the defaults which will be calculated from the transmission speed.

Default for this option is dependant on the transmission speed.

'CD=CHECKCARRIER/K'

For a dial-up connection that can fall apart any second you'll want the driver to periodically look at whether the link layer is still doing something useful. If you choose 'CHECKCARRIER=ON', the driver will go offline as soon as the remote drops the connection. 'CHECKCARRIER=OFF' will cause the driver to ignore the absence of a carrier signal. This option is ignored if you are using the 'NULLMODEM=ON' option.

Default for this option is 'CHECKCARRIER=ON'.

'DIAL/K'

This is where you provide the dial command for the modem. The command can include special escape characters, as were described for the 'INIT/K' parameter. The dial command will be sent after the initialization command, and the modem must respond to it with 'CONNECT' for the protocol initialization to begin. A dial command can look like 'ATD0191011\r'.

Default for this option is an empty string, i.e. no dial command is sent, which is not particularly helpful, yow!

'DIALTIMEOUT/K/N'

The dialer won't wait forever for the modem to respond. At some point of time it will consider the dial attempt failed and abort. The same timeout is used both for the modem initialization and the dial command.

Default for the dial timeout is 60 seconds.

'EOF/K'

If the serial driver supports it, you can enable the so-called *EOF mode* which allows the PPP driver to read complete data frames off the wire without having to assemble them from pieces of the incoming data stream. This can result in improved data throughput, but need not. My tests so far suggest that the improvement is marginal at best. Take care, some serial drivers do not properly support this feature. You should test this before making the option permanent.

Default for this option is 'EOF=OFF'.

'HANGUP/K'

This is where you provide the hangup command for the modem. The command can include special escape characters, as were described for the INIT/K parameter. The hangup command will be sent before the link device is closed.

Default for this option is '~~~+++~~~ATH0\r'.

'IGNOREFCS/K'

TCP/IP protects segments in transfer by end-to-end checksums, which makes the checksum in PPP frames largely redundant. Yet, you're on the safe side if you keep them enabled. However, for PPP over ISDN each HDLC frame transferred (which has a PPP frame in it) you might be able to assume that because the link level device delivered the frame, it must be in good order, and ignore the PPP frame checksum. This can yield a little extra speed. Don't expect any miracles, though.

Default for this option is 'IGNOREFCS=OFF'.

'INIT/K'

This is where you provide the initialization command for the modem. The command can include special escape characters, as follows:

'^@ .. ^_'

To produce a control character, use the caret. For example, '^M' will produce a carriage return character and '^N' will produce a line feed. To produce a caret, use '^^'.

'\a' Produces a bell character (ASCII 5).

'\b' Produces a backspace character (ASCII 8).

'\t' Produces a tabulator character (ASCII 9).

'\n' Produces a line feed character (ASCII 10).

'\f' Produces a form feed character (ASCII 12).

'\r' Produces a carriage return character (ASCII 13).

'\\' Produces a backslash character (ASCII 92).

'\^' Produces a caret character (ASCII 94)

'\~' Produces a tilde character (ASCII 126)

'\xNN' Produces a character, where NN must be in hexadecimal notation. For example, '\xA' will produce a line feed character and '\xDF' will produce the letter 'ß'.

'\YYY' Produces a character, where YYY must be in octal notation. For example, '\15' produces the carriage return character and '\304' produces the letter 'Ä'.

'~' Waits for half a second.

These can be combined into a command string to initialize a modem, such as 'AT&F&D2Bo70&C1&D1\r'. When sent to the modem, the modem must respond with 'OK' to the command. A response of 'ERROR' will cause the dialer to abort.

Default for this option is an empty string, i.e. no initialization command is sent.

'NULLMODEM/K'

If you don't want to connect to an ISP, test the driver or just hook up the Amiga to your Linux box, use a nullmodem cable and tell the driver not to do anything silly, such as trying to dial out, by specifying the 'NULLMODEM=ON' option.

Default for this option is 'NULLMODEM=OFF'.

'PFC/K'

This controls a PPP feature by the name of Protocol-Field-Compression, which is really just a fancy name for a technique that can save one single byte on every IP packet transmitted. If you don't want to save that byte, use 'PFC=OFF'.

Default for this option is 'PFC=ON'.

'READREQUESTS/K/N'

With the so-called *EOF mode* enabled, the driver can queue several read requests to pick up incoming data frames. Here is how you can control how many read requests should be prepared for this task. This parameter also has an effect on how many data frames the driver will expect to arrive per second if it is not operating in *EOF mode*. The higher you set this value, the more incoming data can be processed.

Default for this option is '8'.

'SHARED/K'

You can use a different dialer to make the connection to the remote and then fire off `ppp-serial.device` to establish the PPP connection by having dialer and device driver share the same serial device. To do that, use the 'SHARED=ON' option.

Default for this option is 'SHARED=OFF'.

'VJHC/K'

This controls a PPP feature called *Van Jacobson header compression*, which can shorten IP packets during transmission by theoretically up to 125 bytes each (best case). In my test environment I found that on the average packets shrunk by about 48-49 bytes. If you don't want your IP headers to be compressed, use 'VJHC=OFF'.

A few words on how helpful Van Jacobson header compression actually is. You might think that at all times it's a good thing to have header compression enabled because it reduces the amount of data that has be transmitted for each TCP/IP packet sent.

This is, contrary to what you might have expected, not true. Van Jacobson compression was developed for slow lines (2400 - 19200 bits per second) on which every redundant byte sent has a significant impact on the responsiveness of the connection. This is typically felt with the 'telnet' application which can behave sluggishly when typing single letters.

The Van Jacobson compression scheme also assumes that only a small number of TCP/IP channels will be active at a time (in this implementation, up to 16). If you have a fast link, then you will gain nothing from Van Jacobson compression. If more than a small number of TCP/IP channels are active at a time, which is typically the case with HTTP, then Van Jacobson compression will not be helpful. Worse, because the channel table will keep overflowing, the decompression code will have to toss valid packets to make room for the new connections.

This can have a significant impact on transmission speed, especially with HTTP 1.1 streaming and maximum size PPP packets. Think it over and choose carefully!

Default for this option is '`VJHC=ON`'.

7.1.4 IP packet filter configuration files in "S:IPF"

Amiga-specific examples for the `ipf.rules` and `ipnat.rules` files are stored in the `S:IPF` drawer. A script file `Start-Firewall`, which has the effect of loading the configuration files into the IP packet filter, is stored in the `S:` drawer.

What goes into these files, and how the rules work together is described in the documentation files `ipf-howto.html`, `ipf-rules.doc` and `ipnat-rules.doc`. These go into great detail to explain how you might write or modify your own firewall rules.

The example files `ipf.rules` and `ipnat.rules` assume that you will be connecting your Amiga to the Internet through a network interface called `PPPoE` and that at the time you activate the IP packet filter, the `PPPoE` network interface is already operational.

7.1.4.1 Activating the IP packet filter

The example script stored in `S:Start-Firewall` contains the following instructions:

```
ipf -DFa
ipf -f S:IPF/ipf.rules -E
ipnat -CF -f S:IPF/ipnat.rules
```

The respective commands will do the following:

'`ipf -DFa`'
: This will disable the IP filter (-D) and then discard any currently active IP filter rules (-Fa). Effectively, this will disrupt the firewall operations and disable all currently active connections which use the firewall. It also has the effect of starting with a clean slate for the next following configuration command.

'`ipf -f S:IPF/ipf.rules -E`'
: This will read the IP packet filter rules from the file `S:IPF/ipf.rules` (-f S:IPF/ipf.rules) and subsequently enable the filter (-E).

'`ipnat -CF -f S:IPF/ipnat.rules`'
: This will first delete all NAT rules (-C) and the NAT translation information currently active (-F), and then load the new NAT rules from the file `S:IPF/ipnat.rules` (-f S:IPF/ipnat.rules). This has the effect of starting with a clean slate when the new rules are loaded.

If these commands succeed, then the new set of IP filtering and NAT rules will be loaded and active.

Using the provided example rule files, your Amiga will become a gateway router with a real firewall which other computers in the same network can use to connect to the Internet. All you need to do is configure these computers to use your Amiga's IP address as the default route (or default gateway) address.

Note that in order for the firewall functionality to be really useful, you must have two different Ethernet cards in your Amiga: one which connects to the Internet, and the other connecting to the local network. Never use the same Ethernet card to connect both to the Internet and the local network, as this will nullify the effect of the firewall.

7.1.4.2 Deactivating the IP packet filter

The example script stored in `S:Stop-Firewall` contains the following instructions:

```
ipf -DFa
ipnat -CF
```

The respective commands will do the following:

'`ipf -DFa`'
: This will disable the IP filter (-D) and then discard any currently active IP filter rules (-Fa). Effectively, this will disrupt the firewall operations and disable all currently active connections which use the firewall.

'`ipnat -CF`'
: This will delete all NAT rules (-C) and the NAT translation information currently active (-F).

When these commands have completed their tasks, then IP filter will have been completely disabled, and any resources previously assigned to keeping track of filtering or network address translation will have been released.

7.1.4.3 The IP filtering rules

The example file `ipf.rules`, as stored in the `S:IPF` drawer, imposes rules on the traffic that might pass through your Amiga, and into the local network through the Amiga.

This example assumes that you will be connecting to the Internet through a network interface called `PPPoE`, and that your local network is using the 192.168.0.0/24 address range:

```
#
#  This is an example of a fairly heavy firewall used to keep everyone
#  out of a particular network while still allowing people within that
#  network to get outside.
#
#  The example assumes it is running on a gateway with interface "PPPoE"
#  attached to the outside world, and interface "Ethernet" attached to
#  network 192.168.0.0 which needs to be protected.
#
#
#  Pass any packets not explicitly mentioned by subsequent rules
#
pass out from any to any
pass in from any to any
pass out proto tcp all keep state
#
#  Block any inherently bad packets coming in from the outside world.
#  These include ICMP redirect packets, IP fragments so short the
#  filtering rules won't be able to examine the whole UDP/TCP header,
#  and anything with IP options.
#
block in log quick on PPPoE proto icmp from any to any icmp-type redir
block in log quick on PPPoE proto tcp/udp all with short
block in log quick on PPPoE from any to any with ipopts
#
#  Block any IP spoofing atempts.  (Packets "from" our network
#  shouldn't be coming in from outside).
#
block in log quick on PPPoE from 192.168.0.0/24 to any
block in log quick on PPPoE from localhost to any
block in log quick on PPPoE from 0.0.0.0/32 to any
block in log quick on PPPoE from 255.255.255.255/32 to any
#
#  Block all incoming UDP traffic except talk and DNS traffic.  NFS
#  and portmap are special-cased and logged.
#
```

```
block in on PPPoE proto udp from any to any
block in log on PPPoE proto udp from any to any port = sunrpc
block in log on PPPoE proto udp from any to any port = 2049
pass in on PPPoE proto udp from any to any port = domain
#
#  Allow incoming TCP connections to ports between 1024 and 5000, as
#  these don't have daemons listening but are used by outgoing
#  services like ftp and talk.
#
pass in on PPPoE proto tcp from any to any port 1024 >< 5000
#
#  Block all incoming TCP traffic connections to known services,
#  returning a connection reset so things like ident don't take
#  forever timing out.  Don't log ident (auth port) as it's so common.
#
block return-rst in log on PPPoE proto tcp from any to any flags S/SA
block return-rst in log on PPPoE proto tcp from any to any port = auth flags S/SA
block return-rst in log on PPPoE proto tcp from any to any port = 137
block return-rst in log on PPPoE proto tcp from any to any port = 138
block return-rst in log on PPPoE proto tcp from any to any port = 139
#
#  Now allow various incoming TCP connections to particular hosts, TCP
#  to the main nameserver so secondaries can do zone transfers, SMTP
#  to the mail host, www to the web server (which really should be
#  outside the firewall if you care about security), and ssh to a
#  hypothetical machine called 'gatekeeper' that can be used to gain
#  access to the protected network from the outside world.
#
#  These lines are commented out and should be used only as references.
#
#pass in on PPPoE proto tcp from any to ns port = domain
#pass in on PPPoE proto tcp from any to mail port = smtp
#pass in on PPPoE proto tcp from any to www port = www
#pass in on PPPoE proto tcp from any to gatekeeper port = ssh
```

What these particular rules do, and what other rules you might want to use, is explained in great detail in the `ipf-rules.doc` file.

If you want to experiment with the rule file and make changes, you should always test whether the rule file contents are correct and to not contain typos preventing them from being processed properly.

This test can be performed by the script file stored under `S:Check-Firewall-Rules`. If there is an issue with the rules, then the script will print an error message and exit. Note that the test will not modify your current firewall settings, and is thus safe to use at any time.

To activate your changes, please use the `S:Start-Firewall` script.

7.1.4.4 The network address translation (NAT) rules

The example file `ipnat.rules`, as stored in the `S:IPF` drawer, tells the IP filter to translate the address information stored in packets leaving/entering into the local network through your Amiga.

This example assumes that you will be connecting to the Internet through a network interface called `PPPoE`, and that your local network is using the 192.168.0.0/24 address range:

```
# Scenario: Two network interfaces; one connected to internal 192.168.0.XXX
# network, other connected externally to the Internet. Suppose the internal
# interface is named "Ethernet" and the external interface is named "PPPoE". The
# following mapping will provide the internal network with Internet
# connectivity for tcp/udp traffic (note the "Ethernet" name is not used; instead
# its network address is used):
map PPPoE 192.168.0.0/24 -> PPPoE/32 proxy port ftp ftp/tcp
map PPPoE 192.168.0.0/24 -> PPPoE/32 portmap tcp/udp 10000:20000
map PPPoE 192.168.0.0/24 -> PPPoE/32
```

What these particular rules do, and what other rules you might want to use, is explained in great detail in the `ipnat-rules.doc` file.

If you want to experiment with the rule file and make changes, you should always test whether the rule file contents are correct and to not contain typos preventing them from being processed properly.

This test can be performed by the script file stored under `S:Check-Firewall-Rules`. If there is an issue with the rules, then the script will print an error message and exit. Note that the test will not modify your current firewall settings, and is thus safe to use at any time.

To activate your changes, please use the `S:Start-Firewall` script.

7.2 Shell commands

7.2.1 Configuration

7.2.1.1 AddNetInterface

```
NAME
     AddNetInterface - Make network interfaces known to the protocol stack.

FORMAT
     AddNetInterface [QUIET] [TIMEOUT=<n>] INTERFACE

TEMPLATE
     INTERFACE/M,QUIET/S,TIMEOUT/K/N

PATH
     C:ADDNETINTERFACE

FUNCTION
     ADDNETINTERFACE starts the specified network interfaces, thus starting
     the connection.

OPTIONS
     INTERFACE/M
         The name of the interface to add; this can be a plain interface
         name, such as "Ariadne", or the fully qualified file name which
         contains the interface configuration information. The tool
         expects the name of the file in question (without the prefixed
         path) to become the name of the interface. For historic reasons
         interface names cannot be longer than 15 characters.

         For your convenience, a wild card pattern can be specified in
         place of the file name to use.

         If several interface names are specified, they will be sorted in
         alphabetical order before they are added. If the interface
         files have icons attached, you can use tool types such as
         "PRI=5" or "PRIORITY=5" to select the order in which the interfaces
         will be sorted. Higher priority entries will appear before lower
         priority entries. If the priorities for two entries is identical,
         then the interface names will be compared. If no priority is
         given, the value 0 will be used.

     QUIET/S
         This option causes the program not to emit any error messages
         or progress reports. Also, if the program encounters an error
         it will flag this as failure code 5 which can be looked at
         using the "if warn" shell script command. If this option is
         not in effect, failure codes will be more severe and all sorts
         of progress information will be displayed.

     TIMEOUT/K/N
         If you're going to use DHCP configuration for any of the
         interfaces, a default timeout value of 10 seconds will
         limit the time an interface can take to be configured.
         This parameter allows you to use a different timeout value.
         Note that due to how the configuration protocol works,
         the timeout cannot be shorter than ten seconds.

     The 'AddNetInterface' command can be invoked from Workbench, too. It
     operates on the same configuration files with the same keywords, etc.
     To make it work, create an icon for your interface configuration file
     (it must be a project icon) and put 'AddNetInterface' into its default
     tool. Make sure that the project has enough stack space assigned (4000
```

```
     bytes minimum), then double-click on the icon. If things should go
     wrong, you will see an error requester pop up, and no further
     initialization will be done. You can configure two options in the
     project file's tool types: QUIET and TIMEOUT. These are identical to
     the two parameters of the same name you could pass on the command
     line; they define whether the command should print any error messages
     (the default is to print them) and how long the command should wait
     for DHCP configuration to conclude (default is a timeout of 10
     seconds).

NOTES
     This command is similar to the Unix "ifconfig" command.

     The program makes two passes over the configuration files to be
     taken into account. In the first pass information is gathered
     on the interfaces to add, which is subsequently used to add those
     interfaces found. In the second pass interfaces are configured,
     setting their IP addresses, etc. If anything goes wrong in the
     first pass, processing will stop and no second pass will be
     done. If anything goes wrong in either the first or the second
     pass, that pass will not be completed.

CONFIGURATION FILES
     Interfaces are configured through files stored in the
     "DEVS:NetInterfaces" or "SYS:Storage/NetInterfaces" directories.
     These are text files whose contents are described below.

     Each line of the file must correspond to an option; if a line is
     introduced by a '#' or ';' character it will be ignored (so are empty
     lines). The following options are supported:

         DEVICE/K
             Must be provided; the name of the SANA-II device driver. This
             should be the complete, fully qualified path to the driver. If
             no complete path is provided, the 'Devs:Networks' drawer will be
             checked. Thus, "DEVS:Networks/ariadne.device" is equivalent to
             "ariadne.device".

         UNIT/K/N
             Unit number of the device driver to open. The default is to
             use unit 0.

         IPTYPE/K/N
             You can use this parameter to override the packet type the
             stack uses when sending IP packets; default is 2048 (for
             Ethernet hardware).

         ARPTYPE/K/N
             You can use this parameter to override the packet type the
             stack uses when sending ARP packets. Default is 2054; this
             parameter only works with Ethernet hardware and should not be
             changed.

         IPREQUESTS/K/N
             The number of IP read requests to allocate and queue for the
             SANA-II device driver to use. The default value is 32, larger
             values can improve performance, especially with fast device
             drivers.

         WRITEREQUESTS/K/N
             The number of IP write requests to allocate and queue for the
             SANA-II device driver to use. The default value is 32, larger
             values can improve performance, especially with fast device
             drivers.
```

```
ARPREQUESTS/K/N
    The number of ARP read requests to allocate and queue for the
    SANA-II device driver to use. The default value is 4.

DEBUG/K  (possible parameters: YES or NO)
    You can enable debug output for this interface (don't worry,
    you can always disable it later) to help in tracking down
    configuration problems. At this time of writing, the debug
    mode will, if enabled, produce information on the progress of
    the DHCP configuration process.

POINTTOPOINT/K  (possible parameters: YES or NO)
    This indicates that the device is used for point to point
    connections. The stack automatically figures out whether the
    SANA-II device driver is of the point to point type, so you
    should not need to specify this option.

MULTICAST/K  (possible parameters: YES or NO)
    This tells the stack that this device can handle multicast
    packets. 'YES' only works with Ethernet hardware (where it's
    enabled by default anyway).

DOWNGOESOFFLINE/K  (possible parameters: YES or NO)
    This option is useful with point to point devices, like
    'ppp.device'. When specified, bringing the interface 'down'
    (via the 'ConfigureNetInterface' program) or shutting down the
    stack will cause the associated SANA-II device driver to be
    switched offline (via the 'S2_OFFLINE' command).

REPORTOFFLINE/K  (possible parameters: YES or NO)
    When a device is switched offline, you may want to know about
    it. This is helpful with SLIP/PPP connections which run over a
    serial link which accumulates costs while it is open. When the
    connection is broken and the device goes offline, you will
    receive a brief notification of what happened. However, if you
    tell the library itself to shut down, no notification that a
    device was switched offline will be shown.

REQUIRESINITDELAY/K  (possible parameters: YES or NO)
    Some devices need a little time to settle after they have been
    opened or they will hickup and lose data after the first
    packet has been sent. The original 'Ariadne I' card is one
    such device. For these devices, the 'REQUIRESINITDELAY=YES'
    option will cause a delay of about a second before the first
    packet is sent.

    This option defaults to YES.

COPYMODE/K  (possible parameters: SLOW or FAST)
    This option is for chasing subtle bugs in the driver interface
    with cards like the original 'Ariadne I'. Cards like these do
    not support writing to the hardware transmit buffer in units
    other than 16 bits a piece. Default is 'SLOW', which is
    compatible with the Ariadne I. But if you're feeling
    adventurous, try the 'FAST' option (and don't complain if it
    doesn't work for you!).

FILTER/K  (possible parameters: OFF, LOCAL, IPANDARP or EVERYTHING)
    This option enables the use of the Berkeley packet filter for
    this particular interface. Possible choices for the key are:

        FILTER=OFF
            Disables the filter.

        FILTER=LOCAL
```

```
                Enables filtering on all IP and ARP packets that are
                intended for this particular interface. Packets
                intended for other interfaces or hosts are ignored.

            FILTER=IPANDARP
                Enables filtering on all IP and ARP packets that
                happen to fly by this interface, no matter whether the
                packets are intended for it or not. This requires that
                the underlying network device driver is opened for
                exclusive access in so-called 'promiscuous' mode. This
                may not work if other clients (Envoy, ACS) need to
                keep the driver opened.

            FILTER=EVERYTHING
                Identical to FILTER=IPANDARP, but will also filter all
                other kinds of packets that may show up.

        Default for this option is 'FILTER=LOCAL'. Note that by using
        this option you merely define what the filter mechanism can do
        and what it cannot do. The filter is not enabled when you add
        the interface.

    HARDWAREADDRESS/K
        You can specify the hardware address (layer 2 address, MAC
        address) this interface should respond to when it is first
        added and configured. This usually works only once for each
        interface, which means that once an address has been chosen
        you have to stick with it until the system is rebooted. And it
        also means that the first program to configure the address
        will manage to make its choice stick.

        The hardware address must be given as six bytes in hexadecimal
        notation, separated by colon characters, like this:

            HARDWAREADDRESS=00:60:30:00:11:22

        Take care, there are rules that apply to the choice of the
        hardware address, which means that you cannot simply pick a
        convenient number and get away with it. It is assumed that you
        will want to configure an IEEE 802.3 MAC address, which works for
        Ethernet hardware and is six bytes (48 bits) in size.

In addition to the purely static interface configuration information you
can also tell the configuration program to do something about the
interfaces once they have all been added. That's when the following
configuration file parameters will be taken into account:

    ADDRESS/K
        This configures the IP address of the interface. The parameter
        you supply should be an IP address in dotted-decimal notation
        ("192.168.0.1"). Don't pick a symbolic host name as the system
        may not yet be in a position to talk to name resolution server
        and translate the symbolic name.

        In place of the IP address you can also specify "DHCP"
        (Dynamic Host Configuration Protocol). As the name suggests,
        this will start a configuration process involving the DHCP
        protocol which should eventually yield the right IP address
        for this host. Note that this configuration procedure only
        works for Ethernet hardware.

    ALIAS/K/M
        In addition to the primary interface address you can assign
        several aliases to it. These must be specified in
        dotted-decimal notation ("192.168.0.1"). Alias addresses are
```

```
    added after the primary interface address has been configured.

STATE/K
    By default, interfaces whose addresses are configured will
    switch automatically to 'up' state, making it possible for the
    TCP/IP stack to use them for network I/O. You can override
    this by using the 'STATE=DOWN' switch. The alternatives
    'online' (implies 'up', but tells the underlying network
    interface driver to go online first) and 'offline' (implies
    'down' but tells the driver to go offline first) are available
    as well.

NETMASK/K
    This selects the subnet mask for the interface, which must be
    specified in dotted-decimal notation ("192.0.168.1").

    In place of the subnet mask you can also specify "DHCP"
    (Dynamic Host Configuration Protocol). As the name suggests,
    this will start a configuration process involving the DHCP
    protocol which should eventually yield the right subnet mask
    for this host. Note that this configuration procedure only
    works for Ethernet hardware.

DESTINATION=DESTINATIONADDR/K
    The address of the point-to-point partner for this interface;
    must be specified in dotted-decimal notation ("192.168.0.1").
    Only works for point-to-point connections, such as PPP.

METRIC/K/N
    This configures the interface route metric value. Default
    is 0.

MTU/K/N
    You can limit the maximum transmission size used by the TCP/IP
    stack to push data through the interface. The interface driver
    will have its own ideas about the maximum transmission size.
    You can therefore only suggest a smaller value than the
    driver's preferred hardware MTU size.

CONFIGURE/K  (possible parameters: DHCP, AUTO or FASTAUTO)
    You can use DHCP configuration for this interface and protocol
    stack internals, namely the list of routers (and the default
    gateway) to use and the domain name servers. This option
    allows you to bring up the complete network configuration in
    one single step.

    You can request that a particular IP address is assigned to
    this interface by the DHCP process by specifying
    CONFIGURE=DHCP and your choice of ADDRESS=xxx.xxx.xxx.xxx.

    If your network has no DHCP server, you may choose
    CONFIGURE=AUTO to use automatic IPv4 address selection,
    based upon a protocol called ZeroConf. This protocol will
    select a currently unused address from a specially
    designated address range.

    If you choose automatic configuration in a wireless network,
    you might want to use CONFIGURE=FASTAUTO instead of
    CONFIGURE=AUTO.

    Note that only the CONFIGURE=DHCP option will attempt to
    set up a default route and a set of DNS servers for you to
    use. The alternatives of CONFIGURE=FASTAUTO and
    CONFIGURE=AUTO are restricted to selecting the network
    interface IPv4 addresses.
```

```
        LEASE/K
            This is a complex option which can be used to request how long
            an IP address should be bound to an interface, via the DHCP
            protocol. Several combinations of options are possible. Here
            is a short list:

                LEASE=300
                LEASE=300seconds
                    This requests a lease of exactly 300 seconds, or
                    five minutes.

                LEASE=30min
                    This requests a lease of 30 minutes.

                LEASE=2hours
                    This requests a lease of 2 hours.

                LEASE=1day
                    This requests a lease of 1 day.

                LEASE=4weeks
                    This requests a lease of 4 weeks.

                LEASE=infinite
                    This requests that the IP address should be
                    permanently bound.

            Blank spaces between the numbers and the qualifiers are
            supported. The qualifiers are tested using substring matching,
            which means for example that "30 minutes" is the same as "30
            min" and "30 m".

            Note that the requested lease time may be ignored by the DHCP
            server. After all, it is just a suggestion and not an order.

        ID/K
            This option works along with the CONFIGURE=DHCP process. It
            can be used to tell the DHCP server by which name the local
            host should be referred to. Some DHCP servers are on good
            terms with their local name resolution services and will add
            the name and the associated IP address to the local host
            database. The name you can supply here cannot be longer than
            255 characters and must be at least 2 characters long. Keep it
            brief: not all DHCP servers have room for the whole 255
            characters.

        DHCPUNICAST/K
            Some DHCP servers may not be able to respond to requests for
            assigning IP addresses unless the responses are sent directly
            to the computer which sent the requests. In such cases you
            might want to use DHCPUNICAST=YES option.

    Unsupported keywords in the configuration file (or typos) will be
    reported, along with the name of the file and the line number.

    The name of the configuration file defines the name of the respective
    interface. Interface names must be unique, and the case of the names
    does not matter. For historic reasons interface names cannot be longer
    than 15 characters. Beyond this no restrictions on naming conventions
    apply.

DHCP PROTOCOL
    A few words on DHCP (Dynamic Host Configuration Protocol). First, it
    only works for Ethernet hardware, so please don't try it with PPP or
```

```
     SLIP. Now it gets a bit technical. Unless you request an address to be
     permanently assigned, DHCP will assign addresses only for a limited
     period of time. This is called a 'lease'. Once an IP address has been
     assigned through DHCP, the lease will be repeatedly extended. The DHCP
     server may over time decide not to extend the lease or assign a new IP
     address to the interface. To stop the lease from getting extended over
     and over again, you must either change the interface's primary IP
     address or mark it 'down'. The library will make a brave attempt to
     get a DHCPRELEASE datagram out to notify the server that the
     previously allocated IP address is no longer in use. Don't count on it
     to work, though. First, the protocol stack might be going down so fast
     that it cannot get the datagram out. Second, when you mark an
     interface 'down' you will effectively pull it out of circulation, it
     will not send any further datagrams. Third, DHCP rides on UDP whose
     second name is 'unreliable datagram protocol', meaning that any
     datagram may get lost or corrupted and nobody will hear about it; this
     is rather hard on DHCP since the release message is sent only once.
     Don't worry. Unless you request permanent leases, the leases will
     eventually time out and the now unused IP address will finally return
     to the pool of addresses available for allocation.

EXAMPLES
     Start the interface called "DSL" and run quietly.

         1> AddNetInterface DSL QUIET

     An example configuration file for the "Ariadne" interface, with
     some options commented out:

         1> Type Devs:NetInterfaces/Ariadne
         device=ariadne.device
         unit=0
         #iprequests=64
         #writerequests=64
         copymode=fast
         #configure=dhcp
         address=192.168.0.1
         netmask=255.255.255.0
         #alias=192.168.0.9
         #hardwareaddress=00:60:30:00:11:22
         #id=a3000ux
         #debug=yes
         #filter=everything

SEE ALSO
     ConfigureNetInterface
     NetShutdown
```

7.2.1.2 AddNetRoute

```
NAME
     AddNetRoute - Add message routing paths.

FORMAT
     AddNetRoute [QUIET] [DESTINATION=<IP>] [HOSTDESTINATION=<IP>]
                 [NETDESTINATION=<IP>] [GATEWAY=<IP>] [DEFAULTGATEWAY=<IP>]

TEMPLATE
     QUIET/S,DST=DESTINATION/K,HOSTDST=HOSTDESTINATION/K,
     NETDST=NETDESTINATION/K,VIA=GATEWAY/K,DEFAULT=DEFAULTGATEWAY/K

PATH
     C:ADDNETROUTE

FUNCTION
```

```
     ADDNETROUTE allows to define routes to hosts or networks via an
     interface.

OPTIONS
     QUIET/S
         This option causes the program not to emit any error messages
         or progress reports. Also, if the program encounters an error
         it will flag this as failure code 5 which can be looked at
         using the "if warn" shell script command. If this option is
         not in effect, failure codes will be more severe and all sorts
         of progress information will be displayed.

     DST=DESTINATION/K
         The destination address of a route (or in other words, where
         the route to be added leads to). This must be an IP address
         or a symbolic name. Some routes may require you to specify
         a gateway address through which the route has to pass.
         Depending upon the address you specify, the protocol stack
         will attempt to figure out whether the destination is
         supposed to be a host or a network.

     HOSTDST=HOSTDESTINATION/K
         Same as the "DST=DESTINATION/K" parameter, except that the
         destination is assumed to be a host (rather than a network).

     NETDST=NETDESTINATION/K
         Same as the "DST=DESTINATION/K" parameter, except that the
         destination is assumed to be a network (rather than a host).

     VIA=GATEWAY/K
         This parameter complements the route destination address;
         it indicates the address to which a message should be sent
         for it to be passed to the destination. This must be an IP
         address or a symbolic name.

     DEFAULT=DEFAULTGATEWAY/K
         This parameter selects the default gateway address (which
         must be specified as an IP address or a symbolic host name)
         all messages are sent to which don't have any particular
         other routes associated with them.
         Another, perhaps less misleading name for "default gateway
         address" is "default route".

NOTES
     The command is similar to the Unix "route" command.

     If you use the "DEFAULT=DEFAULTGATEWAY/K" parameter, all
     other destination addresses you may have specified will be
     ignored. Only one of "DESTINATION", "HOSTDESTINATION" or
     "NETDESTINATION" will be used; choose only one. Before you add
     a new default gateway you should delete the old one or you'll
     get an error message instead.

EXAMPLES
     Define a route to the host 192.168.10.12 through a
     gateway at 192.168.1.1

         1> ADDNETROUTE HOSTDESTINATION 192.168.10.12 VIA 192.168.1.1

SEE ALSO
     DeleteNetRoute
```

7.2.1.3 ConfigureNetInterface

```
NAME
```

```
     ConfigureNetInterface - Configure network interface parameters.

FORMAT
     ConfigureNetInterface [QUIET] [TIMEOUT=<n>] INTERFACE

TEMPLATE
     INTERFACE/A,QUIET/S,ADDRESS/K,NETMASK/K,BROADCASTADDR/K,
     DESTINATION=DESTINATIONADDR/K,METRIC/K/N,MTU/K/N,ALIASADDR/K,
     DELETEADDR/K,ONLINE/S,OFFLINE/S,UP/S,DOWN/S,DEBUG/K,COMPLETE/K,
     CONFIGURE/K,LEASE/K,RELEASE=RELEASEADDRESS/S,ID/K,TIMEOUT/K/N,
     DHCPUNICAST/K

PATH
     C:CONFIGURENETINTERFACE

FUNCTION
     CONFIGURENETINTERFACE is used to define how a network interface will
     react and how it will interact with your network.

OPTIONS
     INTERFACE/A
         The name of the interface to be configured. This is a required
         parameter.

     QUIET/S
         This option causes the program not to emit any error messages
         or progress reports. Also, if the program encounters an error
         it will flag this as failure code 5 which can be looked at
         using the 'if warn' shell script command. If this option is
         not in effect, failure codes will be more severe and all sorts
         of progress information will be displayed.

     ADDRESS/K
         The IP address to assign to this interface. This should be
         specified in dotted-decimal notation ("192.168.0.1") and not as
         symbolic name since the system may not be in a state to perform a
         name resolution.

         In place of the IP address you can also specify "DHCP". As the
         name suggests, this will start a configuration process involving
         the DHCP protocol which should eventually yield the right IP
         address for this host. Note that this configuration procedure only
         works for Ethernet hardware.

     NETMASK/K
         The subnet mask to assign to this interface. This must be
         specified in dotted-decimal notation ("192.168.0.1").

         In place of the subnet mask you can also specify "DHCP". As the
         name suggests, this will start a configuration process involving
         the DHCP protocol which should eventually yield the right
         subnet mask for this host. Note that this configuration procedure
         only works for Ethernet hardware.

     BROADCASTADDR/K
         The broadcast address to be used by this interface; must be
         specified in dotted-decimal notation ("192.168.0.1") and only
         works with interfaces that support broadcasts in the first place
         (i.e. Ethernet hardware).

     DESTINATION=DESTINATIONADDR/K
         The address of the point-to-point partner for this interface; must
         be specified in dotted-decimal notation ("192.168.0.1"). Only
         works for point-to-point connections, such as PPP.
```

```
METRIC/K/N
    Route metric value for this interface.

MTU/K/N
    You can limit the maximum transmission size used by the TCP/IP
    stack to push data through the interface. The interface driver
    will have its own ideas about the maximum transmission size.
    You can therefore only suggest a smaller value than the
    driver's preferred hardware MTU size.

ALIASADDR/K
    This adds another address to this interface to respond to. You
    can add as many aliases as you like, provided you don't run out
    of memory.

DELETEADDR/K
    This removes an alias address from the list the interface is to
    respond to.

UP
DOWN
ONLINE
OFFLINE
    This configures the 'line state' of the interface; four states
    are supported:

    UP
        The protocol stack will attempt to transmit messages
        through this interface (even though it might not be
        online yet).

    DOWN
        The protocol stack will no longer attempt to transmit
        messages through this interface (even though it might
        still be online).

    OFFLINE
        The underlying networking device driver is put offline
        and the protocol stack will no longer try to send
        messages through the interface either.

    ONLINE
        An attempt is made to put the underlying networking
        driver online. If that works, then the protocol stack
        will attempt to transmit messages through this
        interface.

DEBUG/K  (possible parameters: YES or NO)
    You can enable debug output for this interface to help in tracking
    down configuration problems. At this time of writing, the debug
    mode will, if enabled, produce information on the progress of the
    DHCP configuration process.

COMPLETE/K  (possible parameters: YES or NO)
    If you configure an interface in several steps, use this parameter
    in the final invocation of the program. It will tell the TCP/IP
    stack that the configuration for this interface is complete. This
    has the effect of causing the static route definition file to be
    reread, if necessary.

RELEASEADDRESS
    If an IP address was dynamically assigned to an interface, this
    switch will tell ConfigureNetInterface to release it. Note that
    you can only release what was previously allocated.
```

```
CONFIGURE/K  (possible parameters: DHCP, AUTO or FASTAUTO)
    You can use DHCP configuration for this interface and protocol
    stack internals, namely the list of routers (and the default
    gateway) to use and the domain name servers. This option allows
    you to bring up the complete network configuration in one
    single step.

    You can request that a particular IP address is assigned to this
    interface by the DHCP process by specifying CONFIGURE=DHCP and
    your choice of ADDRESS=xxx.xxx.xxx.xxx.

    If your network has no DHCP server, you may choose
    CONFIGURE=AUTO to use automatic IPv4 address selection,
    based upon a protocol called ZeroConf. This protocol will
    select a currently unused address from a specially
    designated address range.

    If you choose automatic configuration in a wireless network,
    you might want to use CONFIGURE=FASTAUTO instead of
    CONFIGURE=AUTO.

    Note that only the CONFIGURE=DHCP option will attempt to
    set up a default route and a set of DNS servers for you to
    use. The alternatives of CONFIGURE=FASTAUTO and
    CONFIGURE=AUTO are restricted to selecting the network
    interface IPv4 addresses.

TIMEOUT/K/N
    If you're going to use DHCP configuration for any of the
    interfaces, a default timeout value of 10 seconds will
    limit the time an interface can take to be configured.
    This parameter allows you to use a different timeout value.
    Note that due to how the configuration protocol works,
    the timeout cannot be shorter than ten seconds.

LEASE/K
    This is a complex option which can be used to request how long an
    IP address should be bound to an interface. Several combinations
    of options are possible. Here is a short list:

        LEASE=300
        LEASE=300seconds

            This requests a lease of exactly 300 seconds, or
            five minutes.

        LEASE=30min

            This requests a lease of 30 minutes.

        LEASE=2hours

            This requests a lease of 2 hours.

        LEASE=1day

            This requests a lease of 1 day.

        LEASE=4weeks

            This requests a lease of 4 weeks.

        LEASE=infinite

            This requests that the IP address should be
```

```
                permanently bound.

            Blank spaces between the numbers and the qualifiers are supported.
            The qualifiers are tested using substring matching, which means
            for example that "30 minutes" is the same as "30 min" and "30 m".

            Note that the requested lease time may be ignored by the DHCP
            server. After all, it is just a suggestion and not an order.

        ID/K
            This option works along with the CONFIGURE=DHCP process. It can be
            used to tell the DHCP server by which name the local host should be
            referred to. Some DHCP servers are on good terms with their local name
            resolution services and will add the name and the associated IP
            address to the local host database. The name you can supply here
            cannot be longer than 255 characters and must be at least 2 characters
            long. Keep it brief: not all DHCP servers have room for the whole 255
            characters.

        DHCPUNICAST/K  (possible parameters: YES or NO)
            Some DHCP servers may not be able to respond to requests for
            assigning IP addresses unless the responses are sent directly
            to the computer which sent the requests. In such cases you
            might want to use DHCPUNICAST=YES option.

    NOTES
        The command is similar to the Unix "ifconfig" command.

        If you tell an interface to go online then the program's return
        code will tell you if the command succeeded: a return value of 0
        indicates success (the interface is now online), and a value
        of 5 indicates that it didn't quite work.

        Configuring the address of an interface has two effects: first,
        the interface will be marked as 'up', meaning that the protocol
        stack will attempt to send messages through it when appropriate.
        Second, a direct route to the interface will be established.

    SEE ALSO
        AddNetInterface
```

7.2.1.4 DeleteNetRoute

```
    NAME
        DeleteNetRoute - Delete a message routing path currently in use.

    FORMAT
        DeleteNetRoute [QUIET] [DESTINATION=<ip>] [DEFAULTGATEWAY=<ip>]

    TEMPLATE
        QUIET/S,DST=DESTINATION/K,DEFAULT=DEFAULTGATEWAY/K

    PATH
        C:DELETENETROUTE

    FUNCTION
        The commands removes a route that was defined in your network.

    OPTIONS
        QUIET/S
            This option causes the program not to emit any error messages
            or progress reports. Also, if the program encounters an error
            it will flag this as failure code 5 which can be looked at
            using the "if warn" shell script command. If this option is
            not in effect, failure codes will be more severe and all sorts
```

```
        of progress information will be displayed.

    DST=DESTINATION/K
        The destination address of a route (or in other words, where
        the route to be added leads to) that should be deleted. This
        must be an IP address or a symbolic name.

    DEFAULT=DEFAULTGATEWAY/K
        The default gateway address to be deleted. This must be an
        IP address or a symbolic name.

NOTES
    This command is similar to the Unix "route" command.

    You can try to delete a route that doesn't exist, but it will
    get you an error message instead of failing gracefully.

SEE ALSO
    AddNetRoute
```

7.2.1.5 DeleteNetRoute

```
NAME
    DeleteNetRoute - Delete a message routing path currently in use.

FORMAT
    DeleteNetRoute [QUIET] [DESTINATION=<ip>] [DEFAULTGATEWAY=<ip>]

TEMPLATE
    QUIET/S,DST=DESTINATION/K,DEFAULT=DEFAULTGATEWAY/K

PATH
    C:DELETENETROUTE

FUNCTION
    The commands removes a route that was defined in your network.

OPTIONS
    QUIET/S
        This option causes the program not to emit any error messages
        or progress reports. Also, if the program encounters an error
        it will flag this as failure code 5 which can be looked at
        using the "if warn" shell script command. If this option is
        not in effect, failure codes will be more severe and all sorts
        of progress information will be displayed.

    DST=DESTINATION/K
        The destination address of a route (or in other words, where
        the route to be added leads to) that should be deleted. This
        must be an IP address or a symbolic name.

    DEFAULT=DEFAULTGATEWAY/K
        The default gateway address to be deleted. This must be an
        IP address or a symbolic name.

NOTES
    This command is similar to the Unix "route" command.

    You can try to delete a route that doesn't exist, but it will
    get you an error message instead of failing gracefully.

SEE ALSO
    AddNetRoute
```

7.2.1.6 ManageNetInterfaces

```
NAME
     ManageNetInterfaces - Move network interface configuration files
         between DEVS:NetInterfaces or SYS:Storage/NetInterfaces depending
         upon whether they can be used at system startup time.

FORMAT
     ManageNetInterfaces [INSTALL|CLEANUP|SYNC] [CHECK] [COMMIT]
                         [QUIET] [VERBOSE]
                         [IGNOREDEVICES|IGNORE=<Pattern>]

TEMPLATE
     ACTION,CHECK/S,COMMIT/S,QUIET/S,VERBOSE/S,IGNORE=IGNOREDEVICES/K

PATH
     C:ManageNetInterfaces

FUNCTION
     The network interface configuration files stored in the
     DEVS:NetInterfaces drawer are used by the AddNetInterfaces command
     when the S:User-Startup script is invoked. Any network interfaces
     available at this time are then added and set up.

     Typically, the user will have to pick the network interface
     configuration files which match the networking hardware and its
     device driver software, once Roadshow has been installed.

     The ManageNetInterfaces command can assist here, by automatically
     identifying which network interface configuration files are
     unusable and should go into the SYS:Storage/NetInterfaces drawer,
     and for which network interfaces there is a corresponding network
     device driver available on the system. The latter would be stored
     in the DEVS:NetInterfaces drawer.

     By automatically identifying those interfaces which are not usable
     and which ones could be usable, less effort should be required to
     pick the right interfaces.

OPTIONS
     ACTION
         If this option is omitted, ManageNetInterfaces will show
         the network interface configuration files which could be moved to
         either the SYS:Storage/NetInterfaces or the DEVS:NetInterfaces
         drawer respectively. If there is no such file which could be
         moved, a message to this effect will be printed instead.

         You may use the INSTALL, CLEANUP and SYNC actions to control
         which files should be moved:

            INSTALL
                The INSTALL action focuses on finding network interface
                configuration files stored in SYS:Storage/NetInterfaces
                for which there is a network device driver available.
                If such configuration files are found, they are scheduled
                to be moved into the DEVS:NetInterfaces drawer. This
                action is most useful for installing new network
                configuration files after Roadshow has just been installed.

            CLEANUP
                The CLEANUP action looks for network interface
                configuration files found in the DEVS:NetInterfaces drawer
                which lack a corresponding network device driver. If
                found, the configuration files are scheduled to be moved
                into the SYS:Storage/NetInterfaces drawer so that they
```

```
            are not activated at system startup time.

        SYNC
            The SYNC action performs both the INSTALL and CLEANUP
            operations, moving the respective network interface
            configuration files as needed.

    CHECK/S
        ManageNetInterfaces automatically checks if moving any network
        interface configuration files could not be completed because
        files of the same name exist in both the DEVS:NetInterfaces
        and SYS:Storage/NetInterfaces drawer drawers if you use the
        INSTALL, CLEANUP or SYNC actions.

        If use neither of these actions, no consistency checking is
        performed by default. Use the CHECK option to force the
        consistency checking to be performed.

    COMMIT/S
        By default, ManageNetInterfaces performs none of the changes
        which it shows as being available. These changes will only be
        made if you also use the COMMIT switch with the action you
        want to perform. This allows you to review the changes without
        accidentally making them at the same time.

    QUIET/S
        Use the QUIET switch to reduce the number of messages
        displayed by ManageNetInterfaces. This mostly concerns
        progress and error messages.

    VERBOSE/S
        The VERBOSE switch will enable additional progress information
        and extra information which would otherwise be omitted. For
        example, if ManageNetInterfaces finds no network interface
        configuration files which could be moved, then the VERBOSE
        switch will make it show which files it found and why these
        files need not be moved.

    IGNORE=IGNOREDEVICES/K
        You can tell ManageNetInterfaces to ignore any network
        interface configuration files which make use of certain
        network device drivers, which is useful specifically
        to avoid moving the configuration files for point-to-point
        devices such as slip.device, cslip.device, ppp-serial.device and
        ppp-ethernet.device. These network device drivers need
        special control programs to work and typically cannot be
        set up and used at system startup time.

        The IGNOREDEVICES option expect an AmigaDOS wildcard pattern
        which matches the names of the network device drivers it
        should ignore.

        Example: IGNORE=(ppp-serial|ppp-ethernet|slip|cslip).device

NOTES
    Network device drivers may already be in memory by the time
    the system starts. Otherwise, network device drivers should
    be stored in the DEVS:Networks directory. ManageNetInterfaces
    will look both into DEVS:Networks and DEVS: to find device
    drivers and then examine the network interface configuration
    files in the DEVS:NetInterfaces and SYS:Storage/NetInterfaces
    drawers. This is how ManageNetInterfaces discovers which
    network device drivers are found on disk and in memory and
    for which network interface configuration files there are
    matching device drivers.
```

```
      ManageNetInterfaces can help in narrowing down the list of
      network interface configuration files which may be used to
      set up network devices at system startup time. However, you
      should still take a closer look at what choices were made
      by the ManageNetInterfaces command. You probably know better
      which networking hardware is installed on your system than
      what ManageNetInterfaces will prepare for you. It may store
      configuration files in the DEVS:NetInterfaces drawer which
      should have remained in the SYS:Storage/NetInterfaces
      drawer.

EXAMPLES
      Show which network interface configuration files could be
      installed, to become active when the system starts:

          1> ManageNetInterfaces install
          Available changes:

            In directory "SYS:Storage/NetInterfaces"
              Network interface file = A2065
              Network device driver  = a2065.device
              Change                 = move to DEVS:NetInterfaces
                                     = (Network device driver available)

          Use ManageNetInterfaces COMMIT INSTALL to perform these changes.

      Now enter the command suggested above:

          1> ManageNetInterfaces commit install
          Planned changes:

            In directory "SYS:Storage/NetInterfaces"
              Network interface file = A2065
              Network device driver  = a2065.device
              Change                 = move to DEVS:NetInterfaces
                                     = (Network device driver available)

          All network interfaces files have been moved successfully.

      The next time you start the Amiga, the "A2065" network configuration
      interface will be started for Roadshow to make use of it.


      Show which network interface configuration files have no matching
      network device driver installed. Such network interfaces will
      not be usable by Roadshow.

          1> ManageNetInterfaces cleanup
          Available changes:

            In directory "DEVS:NetInterfaces"
              Network interface file = Ariadne
              Network device driver  = ariadne.device (not available)
              Change                 = move to SYS:Storage/NetInterfaces
                                       (Network device driver not available)

          Use ManageNetInterfaces COMMIT CLEANUP to perform these changes.

      Now enter the command suggested above:

          1> ManageNetInterfaces commit cleanup
          Planned changes:

            In directory "DEVS:NetInterfaces"
```

```
            Network interface file = Ariadne
            Network device driver  = ariadne.device (not available)
            Change                 = move to SYS:Storage/NetInterfaces
                                     (Network device driver not available)

        All network interfaces files have been moved successfully.

    The next time you start the Amiga, the "Ariadne" network interface
    interface will no longer be started and Roadshow will ignore it.


    Examine all network interface configuration files and show
    which ones should be moved, so that you can use the when
    the system starts. Also show the interface configuration
    files which are not currently usable because they lack a
    matching network device driver:

        1> ManageNetInterfaces
        Available changes:

          In directory "SYS:Storage/NetInterfaces"
            Network interface file = A2065
            Network device driver  = a2065.device
            Change                 = move to DEVS:NetInterfaces
                                   = (Network device driver available)

          In directory "DEVS:NetInterfaces"
            Network interface file = Ariadne
            Network device driver  = ariadne.device (not available)
            Change                 = move to SYS:Storage/NetInterfaces
                                     (Network device driver not available)

        Use ManageNetInterfaces COMMIT SYNC or
            ManageNetInterfaces COMMIT INSTALL or
            ManageNetInterfaces COMMIT CLEANUP to perform these changes.

    This shows all the changes that could be made. You could perform
    them with "ManageNetInterfaces commit install", followed by
    "ManageNetInterfaces commit cleanup". Or you could perform all
    changes with "ManageNetInterfaces commit sync".

SEE ALSO
    AddNetInterface
```

7.2.1.7 NetShutdown

```
NAME
    NetShutdown - Attempt to shut down the network in an orderly fashion.

FORMAT
    NetShutdown [TIMEOUT=<secs>] [QUIET]

TEMPLATE
    TIMEOUT/N,QUIET/S

FUNCTION
    The command will stop all running interfaces.

OPTIONS
    TIMEOUT/N
        How many seconds this command should wait until it gives up. By
        default, it will wait up to 5 seconds for the network to shut
        down once it has triggered the shutdown process.

    QUIET/S
```

```
        Use this parameter to stop the command from reporting what it
        is currently doing.

NOTES
     The "NetShutdown" command will trigger the shutdown process of the
     network. This process cannot be stopped once it has started. However,
     this command can make an attempt to wait until the shutdown has
     completed. Normally, the shutdown should be finished in a fraction of
     a second, but at times when other clients still hang onto the network
     resources, the shutdown can fail to complete quite so quickly. In that
     case, the "NetShutdown" command will tell you that it could not
     complete its task within the allocated time frame (within five
     seconds, or whatever timeout you specified). The shutdown, however,
     will proceed and may conclude at a later time.

     When this command starts up it begins by checking if the network is
     currently operational. If this is not the case, it will exit
     immediately, printing a message to this effect.

SEE ALSO
     AddNetInterface
     ShowNetStatus
```

7.2.1.8 RemoveNetInterface

```
NAME
     RemoveNetInterface - Make the protocol stack forget about
         a network interface

FORMAT
     RemoveNetInterface [QUIET] [FORCE] INTERFACE

TEMPLATE
     INTERFACE/K,QUIET/S,FORCE/S

PATH
     C:REMOVENETINTERFACE

FUNCTION
     REMOVENETINTERFACE attempts to shut down the specified network
     interface, so that it may be added again with different parameters.

OPTIONS
     INTERFACE/K
         The name of the interface to shut down. This must be the name
         previously given to ADDNETINTERFACE.

     QUIET/S
         This option causes the program not to emit any error messages
         or progress reports. Also, if the program encounters an error
         it will flag this as failure code 5 which can be looked at
         using the "if warn" shell script command. If this option is
         not in effect, failure codes will be more severe and all sorts
         of progress information will be displayed.

     FORCE/S
         REMOVENETINTERFACE tries not to shut down an interface which
         may still be in use. You can override this with the FORCE
         option which, however, carries the risk that not all the
         resources associated with the network interface may be released
         until you shut down the network with the NETSHUTDOWN command.

EXAMPLES
     Shut down the interface called "DSL", and run quietly.
```

```
        1> RemoveNetInterface DSL QUIET

SEE ALSO
     AddNetInterface
     NetShutdown
```

7.2.1.9 RoadshowControl

```
NAME
     RoadshowControl - Display and change internal configuration options

FORMAT
     RoadshowControl [SAVE] [QUIET] [GET option] [SET option=value]

TEMPLATE
     SAVE/S,QUIET/S,GET/K,SET/K/F

FUNCTION
     Several internal configuration options which determine the behaviour
     of the TCP/IP stack can be changed at run time. This command will
     display/query option values and can be used to change them, too.

OPTIONS
     GET/K
         Determine if a named option is supported and print its current
         value. If the option does not exist, the command will print
         an error message and return with a warning. You can test for
         the warning with the "IF" shell command.

     SET/K
         Change the value of an option.

     SAVE/S
         When combined with the SET option, also save these settings
         permanently so that they will be used the next time you start
         the TCP/IP stack.

     QUIET/S
         If this option is in effect, neither the SET nor the GET options
         will print the current value of an option.

     If you do no specify and option, all known options and their values
     will be printed.

EXAMPLES
     Check if the udp.cksum option exists:

     1> RoadshowControl get udp.cksum
     udp.cksum = 1

     Check if the bpf.bufsize option exists:
     1> RoadshowControl get bpf.bufsize
     bpf.bufsize: Object not found

     Change the tcp.use_mssdflt_for_remote option:
     1> RoadshowControl set tcp.use_mssdflt_for_remote = 1
     tcp.use_mssdflt_for_remote = 1

NOTES
     You really should know what you are doing when you are changing
     internal configuration options to values which are not the default
     settings. The wrong choices may render the TCP/IP inoperable!

OPTIONS
     Here is a brief list of options that may be supported by your
```

```
Roadshow installation.

    bpf.bufsize
        The size of the Berkeley Packet Filter buffer.

    icmp.maskrepl
        Controls if the ICMP layer responds to mask requests.
        This can be 1 (accept) or 0 (ignore).

    icmp.processecho
        Controls if the ICMP layer responds to echo requests.
        This can be 0 (accept), 1 (ignore) or 2 (drop).

    icmp.procesststamp
        Controls if the ICMP layer responds to time stamp requests.
        This can be 0 (accept), 1 (ignore) or 2 (drop).

    if.receive.useclusters
        Data received from a network interface is normally stored
        in a chunk of memory of 2048 bytes in size, which is called
        a cluster. This use of memory can be wasteful, but choosing
        to use less memory may slightly impair performance because
        the inbound data will need to be copied again.
        This option can be 1 (use clusters; this is the default)
    or 0 to use less memory at the expense of performance.

    if.minclustersize
        Whether data received or to be transmitted is stored in
        a single 2048 byte chunk of data or in smaller portions is
        controlled by this threshold value. Smaller threshold values
        will cause more memory to be used, larger threshold values
        may use less memory at the expense of slight performance
        impairment. The default value is 208 bytes.

    ip.defttl
        Controls the default time-to-live value of IP packets
        generated.

    ip.forwarding
        Controls if IP packets may be forwarded or not.
        This can be 1 (forward) or 0 (drop).

    ip.sendredirects
        Controls if ICMP redirect messages may be generated.
        This can be 1 (yes) or 0 (no).

    ip.subnetsarelocal
        Controls if the Internet addresses of directly
        connected hosts should be considered local, or if
        this also applies to hosts on the same subnet.
        This can be 1 (subnets are local) or 0 (they are not).

    task.controller.priority
        Selects the priority at which the network I/O
        controller Task runs. The priority can affect overall
        network performance.
        This must be in the range -128..127. Default is 0.

    tcp.do_rfc1323
        Controls whether or not the TCP extensions for high
        performance (RFC1323) should be enabled or not.
        Specifically, this covers round trip time measurement
        and the TCP window scale option.
        This can be 1 (enable) or 0 (disable).
```

```
            tcp.do_timestamps
                Controls whether or not the round trip time measurement
                feature should be enabled if the tcp.do_rfc1323 option
                is enabled.
                This can be 1 (enable) or 0 (disable).

            tcp.do_win_scale
                Controls whether or not the TCP window scale option
                should be enabled if the tcp.do_rfc1323 option
                is enabled.
                This can be 1 (enable) or 0 (disable).

            tcp.mssdflt
                Controls the default TCP maximum segment size value.

            tcp.recvspace
                Controls the size of the default TCP receive buffer.

            tcp.rttdflt
                Controls the default TCP retransmit time value.

            tcp.sendspace
                Controls the size of the default TCP transmit buffer.

            tcp.use_mssdflt_for_remote
                Controls if the TCP protocol should use a smaller
                maximum segment size value for packets sent to
                hosts which are not in the local network.
                This can be 1 (yes) or 0 (no).

            tcp.random
                Controls which pseudo-random number generator will be
                used, for example for generating the TCP initial send
                sequence number.

                This can be 0 (see Donald Knuth's "The Art of Computer
                Programming, Volume 2: Seminumerical Algorithms" (3rd
                edition), pp. 185-186.) or 1 (see the paper "Xorshift RNGs",
                by George Marsaglia).

                The Marsaglia algorithm promises to be faster, especially
                on 68000 or 68010 machines.

            udp.cksum
                Controls if checksums should be calculated over
                UDP datagrams to be sent and verified for UDP
                datagrams received.
                This can be 1 (yes) or 0 (no).

            udp.recvspace
                Controls the size of the default UDP receive buffer.

            udp.sendspace
                Controls the size of the default UDP transmit buffer.

ENVIRONMENT VARIABLES
     It is not necessary to change the TCP/IP stack options every time you
     start it. You can also set up global environment variables with the
     SetEnv command which the TCP/IP stack will check when it is started.

     The environment variable names correspond to the options listed above.
     For example, Roadshow/tcp/do_win_scale corresponds to the
     tcp.do_win_scale option, Roadshow/ip/forwarding corresponds to the
     ip.forwarding option, etc.
```

```
To change an environment variable, you would enter the following
in the shell:

    SetEnv SAVE Roadshow/ip/forwarding 1

Note that your shell may not support the SAVE switch, or might have
problems with environment variable names longer than 30 letters. In
such a case you would have to get by with the MakeDir, Echo and
Copy commands, like this:

    MakeDir ENV:Roadshow
    MakeDir ENV:Roadshow/ip
    Echo >ENV:Roadshow/ip/forwarding 1
    Copy ENV:Roadshow ENVARC:Roadshow ALL QUIET

To remove all the variables set by RoadshowControl, so that they are
no longer active immediately when Roadshow starts up, delete the
ENVARC:Roadshow folder, like this:

    Delete ALL QUIET ENVARC:Roadshow

After the folder has been deleted you might want to restart your
computer.
```

7.2.2 Diagnostics

7.2.2.1 arp

```
NAME
     ARP - Address resolution display and control

FORMAT
     ARP [-a|ALL] [-d|DELETE] [-s|SET] [HOSTNAME <name>]
         [ADDRESS <address>] [TEMP] [PUB|PUBLISH] [PRO|PROXY]
         [{-f|FILE} <file name>] [-n|NONAMES|NUMBERS]

TEMPLATE
     -a=ALL/S,-d=DELETE/S,-s=SET/S,HOSTNAME,ADDRESS,TEMP/S,PUB=PUBLISH/S,
     PRO=PROXY/S,-f=FILE/K,-n=NONAMES/S=NUMBERS/S

FUNCTION
     The ARP command displays and modifies the Internet-to-Ethernet
     address translation tables used by the address resolution protocol.
     With no flags, the program displays the current ARP entry for
     hostname. The host may be specified by name or by number, using
     Internet dot notation.

OPTIONS
     ALL, -a
         The command displays all of the current ARP entries.

     DELETE, -d
         Delete an entry for the host specified with the HOSTNAME
         parameter.

         Example: arp delete hostname

     NONAMES, -n
         Show network addresses as numbers (normally arp attempts to
         display addresses symbolically).

     SET, -s
         Create an ARP entry for a host with a Ethernet address. The
         Ethernet address is given as six hex bytes separated by colons.
         The entry will be permanent unless the TEMP option is given in the
```

```
        command. If the PUB option is given, the entry will be
        "published"; i.e., this system will act as an ARP server,
        responding to requests for hostname even though the host address
        is not its own.

        Example: arp set hostname 00:30:ab:0e:d5:ee temp pub

    FILE, -f
        Causes the file filename to be read and multiple entries to be set
        in the ARP tables. Entries in the file should be of the form

            hostname ether_addr [temp] [pub]

        with argument meanings as given for the SET option.
```

7.2.2.2 CheckRoadshowConfig

```
NAME
    CheckRoadshowConfig - Examine every single configuration file used
        by Roadshow, looking for errors and inconsistencies, then
        suggest how these could be fixed.

FORMAT
    CheckRoadshowConfig [QUIET] [VERBOSE]

TEMPLATE
    QUIET/S,VERBOSE/S

PATH
    C:CheckRoadshowConfig

FUNCTION
    Roadshow makes use of configuration files which can be edited or
    updated while the TCP/IP stack is already active. If possible,
    changes made to these files will result in Roadshow immediately
    rereading them, updating its its internal databases.

    Reading the configuration files may turn up errors, which Roadshow
    will report through its logging interface, this usually being the
    network log output window which opens on the Workbench screen.
    It is easy to miss these error notifications. Roadshow might no
    longer work correctly and detecting the problem which led to this
    state becomes much harder. There is an additional complication in
    that the built-in error checking for the majority of configuration
    files is shallow at best. If errors are found, processing will
    discard what is not in order and proceed to the next configuration
    entry.

    The CheckRoadshowConfig command can be used to check every single
    Roadshow configuration file stored in the "DEVS:Internet",
    "DEVS:NetInterfaces" and "SYS:Storage/NetInterfaces" drawers as
    needed. The files are examined, validated and any errors or
    inconsistencies are printed, if possible with hints on how they
    might be fixed.

    You can use CheckRoadshowConfig to produce a report of which
    problems are found, but you can also use it in script files, like so:

        CheckRoadshowConfig quiet
        If warn
            echo "Roadshow configuration may need updating or repairs."
        EndIf

    The following configuration files are checked:
```

```
        DEVS:Internet/groups
        DEVS:Internet/hosts
        DEVS:Internet/name_resolution
        DEVS:Internet/networks
        DEVS:Internet/protocols
        DEVS:Internet/routes
        DEVS:Internet/servers
        DEVS:Internet/services
        DEVS:Internet/users

    Once theses files have been checked, the network configuration files
    stored in the following drawers will be examined:

        DEVS:NetInterfaces
        SYS:Storage/NetInterfaces

    Every network configuration file found in these drawers will be
    checked. This includes checking if the network device driver for
    each of the configuration files in "DEVS:NetInterfaces" is currently
    available and not missing. This check is important because a missing
    driver means that the respective network interface cannot be
    activated at system startup time.

    NOTE: No such check for available network device drivers is performed
          for the configuration files in "SYS:Storage/NetInterfaces"
          because the AddNetInterface command will not activate them as
          part of the system startup.

OPTIONS
    QUIET
        With this option in effect, CheckRoadshowConfig will produce no
        output other than for serious errors such as caused by the
        libraries it needs being unavailable. This is helpful when
        using CheckRoadshowConfig in a script.

    VERBOSE
        By default, CheckRoadshowConfig will not print any message
        with regard to whether the configuration files examines are in
        good order or may need changes/repairs. If you use the VERBOSE
        option, it will print a short message indicating whether it
        detected any problems or not. The VERBOSE switch works even
        if you use the QUIET switch at the same time.

NOTES
    Roadshow must be active when you use the CheckRoadshowConfig command
    or otherwise its consistency checks cannot be performed.

    Network device drivers may already be in memory by the time
    the system starts. Otherwise, network device drivers should
    be stored in the "DEVS:Networks" directory.

    CheckRoadshowConfig will look both into "DEVS:Networks" and "DEVS:"
    to find device drivers and then examine the network interface
    configuration files in the "DEVS:NetInterfaces" and
    "SYS:Storage/NetInterfaces" drawers. This is how CheckRoadshowConfig
    discovers which network device drivers are found on disk and in
    memory and for which network interface configuration files there are
    matching device drivers.

EXAMPLES
    Examine all configuration files and show any defects which should
    be corrected:

        1> CheckRoadshowConfig
        CheckRoadshowConfig: Network device driver name "moschipeth.device"
```

```
            should be changed to "usbmoschipeth.device"; see line 7 of
            "DEVS:NetInterfaces/MosChip".
            CheckRoadshowConfig: Network device driver "moschipeth.device" not
            found; see line 7 of "DEVS:NetInterfaces/MosChip".

        Same as above, but showing only a summary of the examination:

            1> CheckRoadshowConfig quiet verbose
            You may have to update or repair your Roadshow configuration files.

    SEE ALSO
        AddNetInterface, ManageNetInterfaces
```

7.2.2.3 GetNetStatus

```
    NAME
        GetNetStatus - Query whether the network is operational.

    FORMAT
        GetNetStatus [CHECK=condition[,condition...]] [QUIET]

    TEMPLATE
        CHECK/K,QUIET/S

    FUNCTION
        The command is used to check/display which interfaces are currently
        running and which settings are being configured. It can be used in
        script files or for quick diagnostic purposes.

    OPTIONS
        CHECK/K
            A list of conditions to check, which must be separated by commas.
            the following conditions can be checked:

                INTERFACES
                    Are any networking interfaces configured and operational?

                PTPINTERFACES
                    Are any point-to-point interfaces, e.g. SLIP and PPP,
                    configured and operational?

                BCASTINTERFACES
                    Are any broadcast interfaces, e.g. Ethernet, configured
                    and operational?

                RESOLVER
                    Are any name resolution servers configured?

                ROUTES
                    Is any routing information configured?

                DEFAULTROUTE
                    Is the default route configured?

            If any of the conditions to test for is not satisfied, a message
            to this effect will be printed and the command will exit with
            status 5, which can be tested in script files using the
            'IF WARN' command.

        QUIET/S
            Whatever happens, except for errors no output will be produced.

    NOTES
        If no conditions are to be checked for, then this command will print
        version information and the list of conditions that can be tested
```

```
     for, indicating which ones are satisfied and which are not.
```

7.2.2.4 NetLogViewer

```
ROADSHOW/NETLOGVIEWER                                   ROADSHOW/NETLOGVIEWER

NAME
     NetLogViewer - Capture any debug or notification messages sent by
                    bsdsocket.library or its clients.

FORMAT
     NETLOGVIEWER [CX_POPKEY <Key>] [CX_PRIORITY <Priority>]
                  [CX_POPUP <YES|NO>]

TEMPLATE
     CX_POPKEY/K,CX_PRIORITY/K/N,CX_POPUP/K

FUNCTION
     Normally, any message that bsdsocket.library or its clients produce is
     displayed in a console window. This program will capture and display
     any of these messages in a special window where the messages can be
     reviewed and even saved to disk.

OPTIONS
     CX_POPKEY/K
         Key combination to be pressed in order to show the log message
         window. Default is "shift alt f8".

     CX_PRIORITY/K/N
         Priority this tool's input event handler has in the Commodities
         filter chain. Default is 0, which should not be changed.

     CX_POPUP/K
         Whether or not the log window should appear when the program is
         first started. Defaults to "yes" (use "no" to hide the window)

NOTES
     The "NetLogViewer" program can be started from Workbench, too. In this
     case it will check its icon for tool types whose names and purposes
     match the command template described above.

     You can and should start the "NetLogViewer" program before you add the
     first networking interface (in order to capture any error output).
     Note, however, that if you shut down the bsdsocket.library (by
     using the "NetShutdown" command) the "NetLogViewer" program will exit,
     too.
```

7.2.2.5 ping

```
NAME
     PING - Send ICMP ECHO_REQUEST packets to network hosts

FORMAT
     PING [-c|COUNT <number>] [-d|DEBUG] [-i|INTERVAL <wait>]
          [-l|LOAD <preload>] [-n|NUMERICONLY|NUMERIC] [-o|ONEREPLY]
          [-q|QUIET] [-R|RECORDROUTE] [DONTROUTE] [-s|SIZE <packetsize>]
          [-t|TIMEOUT <seconds>] [-v|VERBOSE] [BELL]
          [HOST] <host name or IP address>

TEMPLATE
     -c=COUNT/K/N,-d=DEBUG/S,-i=INTERVAL/K/N,-l=LOAD/K/N,
     -n=NUMERICONLY/S=NUMERIC/S,-o=ONEREPLY/S,-q=QUIET/S,-R=RECORDROUTE/S,
     DONTROUTE/S,-s=SIZE/K/N,-t=TIMEOUT/K/N,-v=VERBOSE/S,BELL/S,
     HOST/A

FUNCTION
```

```
     PING uses the ICMP protocol's mandatory ECHO_REQUEST datagram to
     elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST
     datagrams (``pings'') have an IP and ICMP header, followed by a
     ``struct timeval'' and then an arbitrary number of ``pad'' bytes used
     to fill out the packet.

OPTIONS
     -c, COUNT
         Stop after sending (and receiving) <number> ECHO_RESPONSE packets.

     -d, DEBUG
         Set the SO_DEBUG option on the socket being used.

     -i, INTERVAL
         Wait <wait> seconds between sending each packet. The default is to
         wait for one second between each packet.

     -l, LOAD
         If <preload> is specified, PING sends that many packets as fast as
         possible before falling into its normal mode of behavior.

     -n, NUMERICONLY, NUMERIC
         Numeric output only.  No attempt will be made to lookup symbolic
         names for host addresses.

     -o, ONEREPLY
         Exit as soon as one reply to a packet sent has been received.

     -q, QUIET
         Quiet output.  Nothing is displayed except the summary lines at
         startup time and when finished.

     -R, RECORDROUTE
         Record route. Includes the RECORD_ROUTE option in the
         ECHO_REQUEST packet and displays the route buffer on returned
         packets. Note that the IP header is only large enough for nine
         such routes. Many hosts ignore or discard this option.

     DONTROUTE
         Bypass the normal routing tables and send directly to a host
         on an attached network. If the host is not on a
         directly-attached network, an error is returned. This option
         can be used to ping a local host through an interface that has
         no route through it.

     -s, SIZE
         Specifies the number of data bytes to be sent. The default is
         56, which translates into 64 ICMP data bytes when combined
         with the 8 bytes of ICMP header data.

     -t, TIMEOUT
         Regardless of how many packets were received, this will make
        ping exit after the given number of seconds have elapsed. Note
         that the timeout value must be > 0.

     -v, VERBOSE
         Verbose output. ICMP packets other than ECHO_RESPONSE that are
         received are listed.

     BELL
         Print a 'bell' control character for each packet received, which
         on the Amiga either flashes the display or plays a sound.

NOTES
     When using ping for fault isolation, it should first be run on the
```

```
     local host, to verify that the local network interface is up and
     running. Then, hosts and gateways further and further away should be
     ‘‘pinged’’. Round-trip times and packet loss statistics are computed.
     If duplicate packets are received, they are not included in the packet
     loss calculation, although the round trip time of these packets is
     used in calculating the minimum/average/maximum round-trip time
     numbers. When the specified number of packets have been sent (and
     received) or if the program is terminated with a SIGINT, a brief
     summary is displayed.

     This program is intended for use in network testing, measurement and
     management. Because of the load it can impose on the network, it is
     unwise to use ping during normal operations or from automated scripts.

ICMP PACKET DETAILS
     An IP header without options is 20 bytes. An ICMP ECHO_REQUEST packet
     contains an additional 8 bytes worth of ICMP header followed by an
     arbitrary amount of data. When a packetsize is given, this indicated
     the size of this extra piece of data (the default is 56). Thus the
     amount of data received inside of an IP packet of type ICMP ECHO_REPLY
     will always be 8 bytes more than the requested data space (the ICMP
     header).

     If the data space is at least eight bytes large, ping uses the first
     eight bytes of this space to include a timestamp which it uses in the
     computation of round trip times. If less than eight bytes of pad are
     specified, no round trip times are given.

DUPLICATE AND DAMAGED PACKETS
     Ping will report duplicate and damaged packets. Duplicate packets
     should never occur, and seem to be caused by inappropriate link-level
     retransmissions. Duplicates may occur in many situations and are
     rarely (if ever) a good sign, although the presence of low levels of
     duplicates may not always be cause for alarm.

     Damaged packets are obviously serious cause for alarm and often
     indicate broken hardware somewhere in the ping packet’s path (in the
     network or in the hosts).

TRYING DIFFERENT DATA PATTERNS
     The (inter)network layer should never treat packets differently
     depending on the data contained in the data portion. Unfortunately,
     data-dependent problems have been known to sneak into networks and
     remain undetected for long periods of time. In many cases the
     particular pattern that will have problems is something that doesn’t
     have sufficient ‘‘transitions’’, such as all ones or all zeros, or a
     pattern right at the edge, such as almost all zeros. It isn’t
     necessarily enough to specify a data pattern of all zeros (for
     example) on the command line because the pattern that is of interest
     is at the data link level, and the relationship between what you type
     and what the controllers transmit can be complicated.

     This means that if you have a data-dependent problem you will probably
     have to do a lot of testing to find it. If you are lucky, you may
     manage to find a file that either can’t be sent across your network or
     that takes much longer to transfer than other similar length files.
     You can then examine this file for repeated patterns that you can test
     using the -p option of ping.

TTL DETAILS
     The TTL value of an IP packet represents the maximum number of IP
     routers that the packet can go through before being thrown away. In
     current practice you can expect each router in the Internet to
     decrement the TTL field by exactly one.
```

```
     The TCP/IP specification states that the TTL field for TCP packets
     should be set to 60, but many systems use smaller values (4.3 BSD uses
     30, 4.2 used 15).

     The maximum possible value of this field is 255, and most Unix systems
     set the TTL field of ICMP ECHO_REQUEST packets to 255. This is why you
     will find you can ``ping'' some hosts, but not reach them with
     telnet or ftp.

     In normal operation ping prints the ttl value from the packet it
     receives. When a remote system receives a ping packet, it can do one
     of three things with the TTL field in its response:

     - Not change it; this is what Berkeley Unix systems did before the
       4.3BSD-Tahoe release.  In this case the TTL value in the received
       packet will be 255 minus the number of routers in the round-trip
       path.

     - Set it to 255; this is what current Berkeley Unix systems do.  In
       this case the TTL value in the received packet will be 255 minus the
       number of routers in the path from the remote system to the pinging
       host.

     - Set it to some other value.  Some machines use the same value for
       ICMP packets that they use for TCP packets, for example either 30 or
       60.  Others may use completely wild values.

BUGS
     Many Hosts and Gateways ignore the RECORD_ROUTE option.

     The maximum IP header length is too small for options like
     RECORD_ROUTE to be completely useful. There's not much that that can
     be done about this, however.
```

7.2.2.6 rsh

```
NAME
     rsh -- remote shell command execution

FORMAT
     rsh [-n] [-l <User name>] [TIMEOUT=<Seconds>] <Host name> <Command>

TEMPLATE
     -n=NULL/S,-l=USERNAME/K,TIMEOUT/K/N,HOST/A,COMMAND/A/F

PATH
     C:rsh

FUNCTION
     The "rsh" command executes shell commands on a remote host. The output of
     the command running on the remote host will be sent to the Amiga shell
     standard output, which also includes error messages sent by the remote
     server (which go to the standard error output).

     The "rsh" command only tells the remote server which command and
     which command options should be used. It is not an interactive remote
     shell, i.e. the AmigaDOS "Break" command or pressing the [Ctrl]+C
     keys in the Amiga shell directly affect the local Amiga "rsh" command
     only. However, if you stop the local Amiga "rsh" command, the connection
     to the remote server will be closed and the remote command will be
     shut down by receiving a termination signal.

OPTIONS
     -n | NULL
         Other than error messages sent by the remote server, command output
```

```
        will not be printed on the Amiga side. The "rsh" command will keep
        waiting for the remote server to send its data, though.

    -l <Remote user name> | USERNAME=<Remote user name>
        The "rsh" command will tell the remote server which user account
        the command to execute will be using. The default is to use the
        local Amiga default user (which for Roadshow might be "root").
        You can override the default user name with this option.

    TIMEOUT
        The maximum number of seconds to wait for the remote command
        command to send its output before exiting. If this timeout
        elapses, the "rsh" command will close the connection to the
        remote server regardless of how much output was received.

    HOST
        The name or IP address of the remote shell server to use.

    COMMAND
        The name of the command, as well as all command line options
        to send to the remote shell server. You may have to enclose
        the entire command line in double quotes.

NOTES
    The "rsh" command will use the shell/tcp service on the remote
    server, which defaults to port number 514.

    The lengths of the local and the remote user names, as transmitted to
    the remote server, are limited to 16 characters. If the respective
    names given are longer than 16 characters, they will be transmitted
    in truncated form.

    The shell service requires only a valid login, and you will
    not be prompted to enter a password. Modern Unix systems generally
    do not enable the shell service by default because of security
    concerns. Some assembly may be required to even get the shell service
    to work in the first place.

    The server may refuse to run the command, and send an error or
    warning message instead. This is not the command output, and it
    will be printed on the standard error stream. Also, the "rsh"
    command will print the message "Remote server has closed
    connection" and then exit with return code 10 (error).

    If the TIMEOUT option is used, and the timeout occurs before the
    remote server has printed all its output, the "rsh" command will
    print the message "Request timed out" and exit with return
    code 5 (warning).

FILES
    DEVS:Internet/services
```

7.2.2.7 SampleNetSpeed

```
NAME
    SampleNetSpeed - Display network I/O performance

FORMAT
    SampleNetSpeed [[INTERFACE] <interface name>] [LEFT <window left edge>]
                   [TOP <window top edge>] [WIDTH <window width>]
                   [HEIGHT <window height>] [SCREEN <name>]

TEMPLATE
    INTERFACE,LEFT/N,TOP/N,WIDTH/N,HEIGHT/N,SCREEN/K
```

```
FUNCTION
     The command can be used to gather statistics on the performance of
     a specific network interface or all network interfaces used by the
     TCP/IP stack. Performance is measured as data throughput when sending
     or receiving data packets to/from the network.

     The performance data is sampled once every second and visualized in
     a window which is updated regularly. In the top half of the window
     the amount of data received is displayed, while the bottom half shows
     the amount of data sent. The sizes of the bars representing the
     performance are always using the same scale. At the bottom of the
     window three figures are printed (from left to right): maximum
     throughput, averaged throughput, last throughput. For each of these
     figures the values are displayed in the form "number of bytes
     received per second / number of bytes sent per second".

OPTIONS
     INTERFACE
         The name of the interface whose performance should be sampled.
         If omitted, information on all networking interfaces will be
         gathered.

     LEFT
     TOP
         These parameters are optional; you can specify the position of
         the window to open. Default is to open it in the top left corner
         of the screen.

     WIDTH
     HEIGHT
         These parameters are optional; you can specify the size of the
         window to open. Default is to open a window that can display a
         300 by 50 pixel area. Note that width and height actually refer
         to the size of the display area and not the total size of the
         window.

     SCREEN
         You can specify the name of a public screen to open the window
         on. If omitted, the window will open on the default public
         screen, which is usually the Workbench screen.

NOTES
     To stop the command, either click on the window close gadget in the
     top left corner or send a break signal to it in the shell.
```

7.2.2.8 traceroute

```
NAME
     TRACEROUTE - print the route packets take to network host

FORMAT
     TRACEROUTE [-m|MAXTTL <ttl>] [-n|NUMERIC] [-p|PORT <number>]
                [-q|QUERIES <number>] [-r|DONTROUTE] [-s|SOURCE <address>]
                [-t|TOS <type>] [-w|WAIT <time>] [-v|VERBOSE] [HOST <name>]
                [PACKETSIZE <size>]

TEMPLATE
     -d=DEBUG/S,-m=MAXTTL/K/N,-n=NUMERIC/S,-p=PORT/K/N,-q=QUERIES/K/N,
     -r=DONTROUTE/S,-s=SOURCE/K,-t=TOS/K/N,-v=VERBOSE/S,-w=WAIT/K/N,
     HOST/A,PACKETSIZE/N

FUNCTION
     The Internet is a large and complex aggregation of network hardware,
     connected together by gateways. Tracking the route one's packets follow
     (or finding the miscreant gateway that's discarding your packets) can be
```

```
     difficult. Traceroute utilizes the IP protocol 'time to live' field and
     attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along
     the path to some host.

     The only mandatory parameter is the destination host name or IP number.
     The default probe datagram length is 38 bytes, but this may be increased
     by specifying a packet size (in bytes) after the destination host name.

OPTIONS
     -m, MAXTTL
         Set the max time-to-live (max number of hops) used in outgoing
         probe packets. The default is 30 hops (the same default used for
         TCP connections).

     -n, NUMERIC
         Print hop addresses numerically rather than symbolically and
         numerically (saves a nameserver address-to-name lookup for each
         gateway found on the path).

     -p, PORT
         Set the base UDP port number used in probes (default is 33434).
         Traceroute hopes that nothing is listening on UDP ports base to
         base+nhops-1 at the destination host (so an ICMP PORT_UNREACHABLE
         message will be returned to terminate the route tracing). If
         something is listening on a port in the default range, this
         option can be used to pick an unused port range.

     -q, QUERIES
         Set the number of probes per ``ttl'' (default is three probes).

     -r, DONTROUTE
         Bypass the normal routing tables and send directly to a host on
         an attached network. If the host is not on a directly-attached
         network, an error is returned. This option can be used to ping a
         local host through an interface that has no route through it.

     -s, SOURCE
         Use the following IP address (which must be given as an IP
         number, not a hostname) as the source address in outgoing probe
         packets. On hosts with more than one IP address, this option can
         be used to force the source address to be something other than
         the IP address of the interface the probe packet is sent on. If
         the IP address is not one of this machine's interface addresses,
         an error is returned and nothing is sent.

     -t, TOS
         Set the type-of-service in probe packets to the following value
         (default zero). The value must be a decimal integer in the range
         0 to 255. This option can be used to see if different
         types-of-service result in different paths. Not all values of TOS
         are legal or meaningful - see the IP spec for definitions. Useful
         values are probably '-t 16' (low delay) and '-t 8' (high
         throughput).

     -v, VERBOSE
         Verbose output. Received ICMP packets other than TIME_EXCEEDED
         and UNREACHABLEs are listed.

     -w, WAIT
         Set the time (in seconds) to wait for a response to a probe
         (default 3 sec.).

DESCRIPTION
     This program attempts to trace the route an IP packet would follow to
     some internet host by launching UDP probe packets with a small ttl
```

```
(time to live) then listening for an ICMP "time exceeded" reply from a
gateway. We start our probes with a ttl of one and increase by one
until we get an ICMP "port unreachable" (which means we got to "host")
or hit a max (which defaults to 30 hops & can be changed with the -m
flag). Three probes (changed with -q flag) are sent at each ttl
setting and a line is printed showing the ttl, address of the gateway
and round trip time of each probe. If the probe answers come from
different gateways, the address of each responding system will be
printed. If there is no response within a 3 sec. timeout interval
(changed with the -w flag), a "*" is printed for that probe.

We don't want the destination host to process the UDP probe packets so
the destination port is set to an unlikely value (if some clod on the
destination is using that value, it can be changed with the -p flag).

A sample use and output might be:

[yak 71]% traceroute nis.nsf.net.
traceroute to nis.nsf.net (35.1.1.48), 30 hops max, 56 byte packet
1  helios.ee.lbl.gov (128.3.112.1)  19 ms  19 ms  0 ms
2  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  39 ms  19 ms
3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  39 ms  19 ms
4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  39 ms  40 ms  39 ms
5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  39 ms  39 ms  39 ms
6  128.32.197.4 (128.32.197.4)  40 ms  59 ms  59 ms
7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  59 ms
8  129.140.70.13 (129.140.70.13)  99 ms  99 ms  80 ms
9  129.140.71.6 (129.140.71.6)  139 ms  239 ms  319 ms
10  129.140.81.7 (129.140.81.7)  220 ms  199 ms  199 ms
11  nic.merit.edu (35.1.1.48)  239 ms  239 ms  239 ms

Note that lines 2 & 3 are the same. This is due to a buggy kernel on
the 2nd hop system - lbl-csam.arpa - that forwards packets with a zero
ttl (a bug in the distributed version of 4.3 BSD). Note that you have
to guess what path the packets are taking cross-country since the
NSFNet (129.140) doesn't supply address-to-name translations for its
NSSes.

A more interesting example is:

[yak 72]% traceroute allspice.lcs.mit.edu.
traceroute to allspice.lcs.mit.edu (18.26.0.115), 30 hops max
1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
2  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  19 ms  19 ms
3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  19 ms
4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  19 ms  39 ms  39 ms
5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  20 ms  39 ms  39 ms
6  128.32.197.4 (128.32.197.4)  59 ms  119 ms  39 ms
7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  39 ms
8  129.140.70.13 (129.140.70.13)  80 ms  79 ms  99 ms
9  129.140.71.6 (129.140.71.6)  139 ms  139 ms  159 ms
10  129.140.81.7 (129.140.81.7)  199 ms  180 ms  300 ms
11  129.140.72.17 (129.140.72.17)  300 ms  239 ms  239 ms
12  * * *
13  128.121.54.72 (128.121.54.72)  259 ms  499 ms  279 ms
14  * * *
15  * * *
16  * * *
17  * * *
18  ALLSPICE.LCS.MIT.EDU (18.26.0.115)  339 ms  279 ms  279 ms

Note that the gateways 12, 14, 15, 16 & 17 hops away either don't send
ICMP "time exceeded" messages or send them with a ttl too small to
reach us. 14 - 17 are running the MIT C Gateway code that doesn't send
"time exceeded"s. God only knows what's going on with 12.
```

```
     The silent gateway 12 in the above may be the result of a bug in the
     4.[23] BSD network code (and its derivatives): 4.x (x <= 3) sends an
     unreachable message using whatever ttl remains in the original
     datagram. Since, for gateways, the remaining ttl is zero, the ICMP
     "time exceeded" is guaranteed to not make it back to us. The behavior
     of this bug is slightly more interesting when it appears on the
     destination system:

     1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
     2  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  39 ms
     3  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  39 ms  19 ms
     4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  39 ms  40 ms  19 ms
     5  ccn-nerif35.Berkeley.EDU (128.32.168.35)  39 ms  39 ms  39 ms
     6  csgw.Berkeley.EDU (128.32.133.254)  39 ms  59 ms  39 ms
     7  * * *
     8  * * *
     9  * * *
     10  * * *
     11  * * *
     12  * * *
     13  rip.Berkeley.EDU (128.32.131.22)  59 ms !  39 ms !  39 ms !

     Notice that there are 12 "gateways" (13 is the final destination) and
     exactly the last half of them are "missing". What's really happening
     is that rip (a Sun-3 running Sun OS3.5) is using the ttl from our
     arriving datagram as the ttl in its ICMP reply. So, the reply will
     time out on the return path (with no notice sent to anyone since
     ICMP's aren't sent for ICMP's) until we probe with a ttl that's at
     least twice the path length. I.e., rip is really only 7 hops away. A
     reply that returns with a ttl of 1 is a clue this problem exists.
     Traceroute prints a "!" after the time if the ttl is <= 1. Since
     vendors ship a lot of obsolete (DEC's Ultrix, Sun 3.x) or non-standard
     (HPUX) software, expect to see this problem frequently and/or take
     care picking the target host of your probes. Other possible
     annotations after the time are !H, !N, !P (got a host, network or
     protocol unreachable, respectively), !S or !F (source route failed or
     fragmentation needed - neither of these should ever occur and the
     associated gateway is busted if you see one). If almost all the probes
     result in some kind of unreachable, traceroute will give up and exit.

     This program is intended for use in network testing, measurement and
     management. It should be used primarily for manual fault isolation.
     Because of the load it could impose on the network, it is unwise to
     use traceroute during normal operations or from automated scripts.

AUTHOR
     Implemented by Van Jacobson from a suggestion by Steve Deering.
     Debugged by a cast of thousands with particularly cogent suggestions
     or fixes from C. Philip Wood, Tim Seaver and Ken Adelman.
```

7.2.2.9 ShowNetStatus

```
NAME
     ShowNetStatus - Display various information about the status of the
                     network configuration.

FORMAT
     ShowNetStatus [INTERFACE=<itf>[,<itf>...]] [INTERFACES] [ARPCACHE=ARP]
     [ROUTES] [DNS=DOMAINNAMESERVERS] [ICMP] [IGMP] [IP] [MB=MEMORY]
     [MR=MULTICASTROUTING] [RT=ROUTING] [TCP] [UDP] [TCPSOCKETS]
     [UDPSOCKETS] [NAMES] [ALL] [REPEAT] [QUIET]

TEMPLATE
     INTERFACE/M,INTERFACES/S,ARPCACHE=ARP/S,ROUTES/S,
```

```
     DNS=DOMAINNAMESERVERS/S,ICMP/S,IGMP/S,IP/S,MB=MEMORY/S,
     MR=MULTICASTROUTING/S,RT=ROUTING/S,TCP/S,UDP/S,TCPSOCKETS/S,
     UDPSOCKETS/S,NAMES/S,ALL/S,REPEAT/S,QUIET/S

PATH
     C:SHOWNETSTATUS

FUNCTION
     This command can display a lot of informations about the active
     interfaces, either for one interface or for all them. It can be used
     to display the current settings or to give details on the quality of
     the network: transfer speed, number of errors, etc.

     You can request which details should be displayed. If you provide
     no options to this command, it will print a general summary of the
     current network status.

OPTIONS
     INTERFACE/M
         For each interface provided in this list, show detailed
         configuration information and statistics.

     INTERFACES/S
         Show the list of all interfaces known to the TCP/IP stack.

     ARP/S
         Show the contents of the address resolution protocol (ARP) cache.

     ROUTES/S
         Show information on the routes that are configured. The
         default route is indicated by having a gateway address
         of 'default'.

     DNS=DOMAINNAMESERVERS/S
         Show a list of domain name servers known to the TCP/IP
         stack. Servers are either dynamically configured or
         statically (through the 'DEVS:Internet/name_resolution' file).

     ICMP/S
         Display Internet Control Message Protocol statistics.

     IGMP/S
         Display Internet Group Management Protocol statistics.

     IP/S
         Display Internet Protocol statistics.

     MB=MEMORY/S
         Display memory buffer statistics.

     MR=MULTICASTROUTING/S
         Display multicast routing statistics.

     RT=ROUTING/S
         Display routing statistics.

     TCP/S
         Display Transmission Control Protocol statistics.

     UDP/S
         Display User Datagram Protocol statistics.

     TCPSOCKETS/S
         Display information about the TCP sockets currently
         in use. Note that unless you use the 'ALL' option,
```

```
            sockets bound to local addresses will not be shown.

        UDPSOCKETS/S
            Display information about the UDP sockets currently
            in use. Note that unless you use the 'ALL' option,
            sockets bound to local addresses will not be shown.

        NAMES/S
            Attempt to figure out which symbolic names are associated
            with the IP addresses to be printed, and also attempt to
            associate port numbers with service names.

        ALL/S
            This option works in conjunction with the TCPSOCKETS and
            UDPSOCKETS parameters. It tells the respective display
            routines to show all currently active sockets, including
            those bound to local addresses.

        REPEAT/S
            Repeat the query and display process each second; the
            screen will be cleared before the new information is
            printed. To stop printing the data, press [Ctrl]+C.

        QUIET/S
            Do not show any error messages.

    EXAMPLES
        Show details on interface "itf".

            1> SHOWNETSTATUS INTERFACE itf

        Show details on the Internet Group Management Protocol and routing
        statistics

            1> SHOWNETSTATUS ROUTING IGMP

    NOTES
        This command works like a combination of the Unix "route", "ifconfig"
        and "netstat" commands.

        Most of the information printed is rather cryptic and not of much
        use to the casual user. It can help in debugging, though.
```

7.2.2.10 tcpdump

Documentation for the `tcpdump` command is provided in a separate file by the name of `tcpdump.doc`.

7.2.3 File transfer

7.2.3.1 ftp

Documentation for the `ftp` command is provided in a separate file by the name of `ftp.doc`.

7.2.3.2 wget

Documentation for the GNU `wget` command is provided in a separate file by the name of `wget.doc`.

7.2.4 IP packet filter

The following section only contains the documentation for the four shell commands which provide for IP packet filter configuration, testing and monitoring. Additional documentation

files `ipf-howto.html`, `ipf-rules.doc` and `ipnat-rules.doc` are provided which go into detail explaining how the commands may be used.

7.2.4.1 ipf

```
NAME
       ipf  -  alters  packet filtering lists for IP packet input
       and output

SYNOPSIS
       ipf [ -AdDEInoPrsvVyzZ ] [ -l <block|pass|nomatch>  ]  [
       -F <i|o|a|s|S> ] -f <filename> [ -f <filename> [...]]

DESCRIPTION
       ipf opens the filenames listed (treating "-" as stdin) and
       parses the file for a set of rules which are to  be  added
       or removed from the packet filter rule set.

       Each rule processed by ipf is added to the kernel's inter-
       nal lists if there are no  parsing  problems.   Rules  are
       added to the end of the internal lists, matching the order
       in which they appear when given to ipf.

OPTIONS
       -A     Set  the  list  to  make changes to the active list
              (default).

       -d     Turn debug mode on.  Causes  a  hexdump  of  filter
              rules to be generated as it processes each one.

       -D     Disable the filter (if enabled).  Not effective for
              loadable kernel versions.

       -E     Enable the filter (if disabled).  Not effective for
              loadable kernel versions.

       -F <i|o|a>
              This  option  specifies which filter list to flush.
              The parameter should either  be  "i"  (input),  "o"
              (output)  or "a" (remove all filter rules).  Either
              a single letter or an entire word starting with the
              appropriate  letter  maybe used.  This option maybe
              before, or after, any other with the order  on  the
              command line being that used to execute options.

       -F <s|S>
              To  flush  entries  from  the  state  table, the -F
              option  is  used  in  conjuction  with  either  "s"
              (removes  state  information  about  any non-fully
              established connections) or "S" (deletes the entire
              state  table).   Only one of the two options may be
              given.  A fully established connection will show up
              in ipfstat -s output as 4/4, with deviations either
              way indicating it  is  not  fully  established  any
              more.

       -f <filename>
              This option specifies which files ipf should use to
              get input from for modifying the packet filter rule
              lists.

       -I     Set  the list to make changes to the inactive list.

       -l  <pass|block|nomatch>
              Use of the -l flag toggles default logging of pack-
```

```
              ets.  Valid  arguments  to  this  option are pass,
              block and nomatch.  When  an  option  is  set,  any
              packet  which  exits  filtering and matches the set
              category is logged.  This is most useful for caus-
              ing all packets which don't match any of the loaded
              rules to be logged.

       -n     This flag (no-change) prevents  ipf  from  actually
              making  any  ioctl  calls  or  doing anything which
              would alter the currently running kernel.

       -o     Force rules by default to be added/deleted  to/from
              the  output  list,  rather than the (default) input
              list.

       -P     Add rules as temporary entries in  the  authentica-
              tion rule table.

       -r     Remove  matching  filter rules rather than add them
              to the internal lists

       -s     Swap the active  filter  list  in  use  to  be  the
              "other" one.

       -v     Turn  verbose mode on.  Displays information relat-
              ing to rule processing.

       -V     Show version information.  This  will  display  the
              version  information  compiled  into the ipf binary
              and retrieve it  from  the  kernel  code  (if  run-
              ning/present).   If  it  is  present in the kernel,
              information about its current state  will  be  dis-
              played  (whether logging is active, default filter-
              ing, etc).

       -y     Manually resync the in-kernel interface list  main-
              tained by IP Filter with the current interface sta-
              tus list.

       -z     For each rule in the input file, reset the  statis-
              tics  for  it  to  zero  and display the statistics
              prior to them being zero'd.

       -Z     Zero global statistics held in the kernel for  fil-
              tering  only (this doesn't affect fragment or state
              statistics).

SEE ALSO
       ipftest(1), mkfilters(1), ipf(4),  ipl(4),  ipf(5),  ipfs-
       tat(8), ipmon(8), ipnat(8)
```

7.2.4.2 ipfstat

```
NAME
       ipfstat  -  reports on packet filter statistics and filter
       list

SYNOPSIS
       ipfstat [ -aAfghIinosv ] [ -d <device> ]

       ipfstat -t [ -C ] [ -D <addrport> ] [ -P <protocol> ] [ -S
       <addrport> ] [ -T <refresh time> ] [ -d <device> ]

DESCRIPTION
       ipfstat  examines  /dev/kmem  using the symbols _fr_flags,
```

```
       _frstats, _filterin, and _filterout.  To run and work,  it
       needs  to  be  able  to read both /dev/kmem and the kernel
       itself.  The kernel name defaults to /vmunix.

       The default behaviour of ipfstat is to retrieve  and  dis-
       play  the  accumulated  statistics which have been accumu-
       lated over time as the kernel has put packets through  the
       filter.

OPTIONS
       -a     Display  the  accounting filter list and show bytes
              counted against each rule.

       -A     Display packet authentication statistics.

       -C     This option is only valid in combination  with  -t.
              Display  "closed"  states  as well in the top. Nor-
              mally, a TCP connection is not  displayed  when  it
              reaches  the  CLOSE_WAIT  protocol state. With this
              option enabled, all state entries are displayed.

       -d <device>
              Use a device other than  /dev/ipl  for  interfacing
              with the kernel.

       -D <addrport>
              This  option  is only valid in combination with -t.
              Limit the state top  display  to  show  only  state
              entries whose destination IP address and port match
              the addport argument. The addrport specification is
              of  the  form  ipaddress[,port].  The ipaddress and
              port should be either numerical or the string "any"
              (specifying  any ip address resp. any port). If the
              -D option is not  specified,  it  defaults  to  "-D
              any,any".

       -f     Show  fragment  state  information (statistics) and
              held state information (in the kernel)  if  any  is
              present.

       -g     Show  groups  currently configured (both active and
              inactive).

       -h     Show per-rule the number of times each one scores a
              "hit".  For use in combination with -i.

       -i     Display  the filter list used for the input side of
              the kernel IP processing.

       -I     Swap between retrieving "inactive"/"active"  filter
              list details.  For use in combination with -i.

       -n     Show  the  "rule  number"  for  each  rule as it is
              printed.

       -o     Display the filter list used for the output side of
              the kernel IP processing.

       -P <protocol>
              This  option  is only valid in combination with -t.
              Limit the state top  display  to  show  only  state
              entries  that  match a specific protocol. The argu-
              ment  can  be  a  protocol  name  (as  defined   in
              /etc/protocols)  or  a  protocol number. If this
              option is not specified, state entries for any pro-
```

```
       tocol are specified.

   -s  Show packet/flow state information (statistics
       only).

   -sl Show held state information (in the kernel) if any
       is present (no statistics).

   -S <addrport>
       This option is only valid in combination with -t.
       Limit the state top display to show only state
       entries whose source IP address and port match the
       addport argument. The addrport specification is of
       the form ipaddress[,port]. The ipaddress and port
       should be either numerical or the string "any"
       (specifying any ip address resp. any port). If the
       -S option is not specified, it defaults to "-S
       any,any".

   -t  Show the state table in a way similar to they way
       top(1) shows the process table. States can be
       sorted using a number of different ways. This
       options requires ncurses(3) and needs to be com-
       piled in. It may not be available on all operating
       systems. See below, for more information on the
       keys that can be used while ipfstat is in top mode.

   -T <refreshtime>
       This option is only valid in combination with -t.
       Specifies how often the state top display should be
       updated. The refresh time is the number of seconds
       between an update. Any postive integer can be used.
       The default (and minimal update time) is 1.

   -v  Turn verbose mode on. Displays more debugging
       information.

SYNOPSIS
   The role of ipfstat is to display current kernel statis-
   tics gathered as a result of applying the filters in place
   (if any) to packets going in and out of the kernel. This
   is the default operation when no command line parameters
   are present.

   When supplied with either -i or -o, it will retrieve and
   display the appropriate list of filter rules currently
   installed and in use by the kernel.

STATE TOP
   Using the -t option ipfstat will enter the state top mode.
   In this mode the state table is displayed similar to the
   way top displays the process table. The -C, -D, -P, -S and
   -T commandline options can be used to restrict the state
   entries that will be shown and to specify the frequency of
   display updates.

   In state top mode, the following keys can be used to
   influence the displayed information:

   d select information to display.

   l redraw the screen.

   q quit the program.
```

```
       s switch between different sorting criterion.

       r reverse the sorting criterion.

       States  can  be sorted by protocol number, by number of IP
       packets, by number of bytes and  by  time-to-live  of  the
       state  entry.  The  default  is  to  sort by the number of
       bytes. States are sorted in descending order, but you  can
       use the r key to sort them in ascending order.

STATE TOP LIMITATIONS
       It  is  currently not possible to interactively change the
       source, destination and protocol  filters  or  the  refreh
       frequency. This must be done from the command line.

       The  screen must have at least 80 columns. This is however
       not checked.

       Only the first X-5 entries that match the sort and  filter
       criteria  are  displayed (where X is the number of rows on
       the display. There is no way to see more entries.

       No support for IPv6


FILES
       /dev/kmem
       /dev/ipl
       /dev/ipstate
       /vmunix

SEE ALSO
       ipf(8)

BUGS
       none known.
```

7.2.4.3 ipmon

```
NAME
       ipmon - monitors /dev/ipl for logged packets

SYNOPSIS
       ipmon  [  -aDFhnpstvxX ] [ -N <device> ] [ -o [NSI] ] [ -O
       [NSI] ] [ -P <pidfile> ] [ -S <device> ] [ -f <device> ] [
       <filename> ]

DESCRIPTION
       ipmon  opens  /dev/ipl  for  reading and awaits data to be
       saved from the packet filter.  The binary data  read  from
       the  device  is  reprinted in human readable for, however,
       IP#'s are not mapped back  to  hostnames,  nor  are  ports
       mapped back to service names.  The output goes to standard
       output by default or a filename, if given on  the  command
       line.   Should  the  -s  option be used, output is instead
       sent to syslogd(8).  Messages sent  via  syslog  have  the
       day, month and year removed from the message, but the time
       (including microseconds), as recorded in the log, is still
       included.

       Messages  generated  by  ipmon consist of whitespace sepa-
       rated fields.  Fields common to all messages are:

       1. The date of packet receipt. This is suppressed when the
       message is sent to syslog.
```

```
       2. The time of packet receipt. This is in the form
       HH:MM:SS.F, for hours, minutes seconds, and fractions of a
       second (which can be several digits long).

       3. The name of the interface the packet was processed on,
       e.g., we1.

       4. The group and rule number of the rule, e.g., @0:17.
       These can be viewed with ipfstat -n.

       5. The action: p for passed or b for blocked.

       6. The addresses. This is actually three fields: the
       source address and port (separted by a comma), the -> sym-
       bol, and the destination address and port. E.g.:
       209.53.17.22,80 -> 198.73.220.17,1722.

       7. PR followed by the protocol name or number, e.g., PR
       tcp.

       8. len followed by the header length and total length of
       the packet, e.g., len 20 40.

       If the packet is a TCP packet, there will be an additional
       field starting with a hyphen followed by letters corre-
       sponding to any flags that were set. See the ipf.conf
       manual page for a list of letters and their flags.
       If the packet is an ICMP packet, there will be two fields
       at the end, the first always being 'icmp', and the next
       being the ICMP message and submessage type, separated by a
       slash, e.g., icmp 3/3 for a port unreachable message.

       In order for ipmon to properly work, the kernel option
       IPFILTER_LOG must be turned on in your kernel. Please see
       options(4) for more details.

OPTIONS
       -a     Open all of the device logfiles for reading log
              entries from. All entries are displayed to the
              same output 'device' (stderr or syslog).

       -D     Cause ipmon to turn itself into a daemon. Using
              subshells or backgrounding of ipmon is not required
              to turn it into an orphan so it can run indefi-
              nately.

       -f <device>
              specify an alternative device/file from which to
              read the log information for normal IP Filter log
              records.

       -F     Flush the current packet log buffer. The number of
              bytes flushed is displayed, even should the result
              be zero.

       -n     IP addresses and port numbers will be mapped, where
              possible, back into hostnames and service names.

       -N <device>
              Set the logfile to be opened for reading NAT log
              records from to <device>.

       -o     Specify which log files to actually read data from.
              N - NAT logfile, S - State logfile, I - normal IP
```

```
              Filter  logfile.   The  -a  option is equivalent to
              using -o NSI.

       -O     Specify which log files you do  not  wish  to  read
              from.   This  is  most  sensibly  used with the -a.
              Letters available as paramters to this are the same
              as for -o.

       -p     Cause  the port number in log messages to always be
              printed as a number and never attempt to look it up
              as from /etc/services, etc.

       -P <pidfile>
              Write  the  pid of the ipmon process to a file.  By
              default this is //etc/opt/ipf/ipmon.pid  (Solaris),
              /var/run/ipmon.pid     (44BSD    or    later)    or
              /etc/ipmon.pid for all others.

       -s     Packet information read in  will  be  sent  through
              syslogd  rather  than saved to a file.  The default
              facility when compiled  and  installed  is  local0.
              The following levels are used:

              LOG_INFO  -  packets logged using the "log" keyword
              as the action rather than pass or block.

              LOG_NOTICE - packets logged which are also passed

              LOG_WARNING - packets logged which are also blocked

              LOG_ERR  - packets which have been logged and which
              can be considered "short".

       -S <device>
              Set the logfile to be opened for reading state  log
              records from to <device>.

       -t     read  the  input  file/device  in  a manner akin to
              tail(1).

       -v     show tcp window, ack and sequence fields.

       -x     show the packet data in hex.

       -X     show the log header record data in hex.

DIAGNOSTICS
       ipmon expects data that it reads to be consistent with how
       it should be saved and will abort if it fails an assertion
       which detects an anomaly in the recorded data.

FILES
       /dev/ipl
       /dev/ipnat
       /dev/ipstate
       /etc/services

SEE ALSO
       ipl(4), ipf(8), ipfstat(8), ipnat(8)
```

7.2.4.4 ipnat

```
NAME
       ipnat - user interface to the NAT
```

```
SYNOPSIS
       ipnat [ -lnrsvCF ] -f <filename>

DESCRIPTION
       ipnat  opens  the filename given (treating "-" as stdin) and parses the
       file for a set of rules which are to be added or removed  from  the  IP
       NAT.

       Each  rule processed by ipnat is added to the kernels internal lists if
       there are no parsing problems.  Rules are  added  to  the  end  of  the
       internal  lists,  matching the order in which they appear when given to
       ipnat.

OPTIONS
       -C     delete all entries in the current NAT rule listing (NAT rules)

       -F     delete all active entries in the current NAT  translation  table
              (currently active NAT mappings)

       -l     Show the list of current NAT table entry mappings.

       -n     This  flag  (no-change)  prevents  ipf  from actually making any
              ioctl calls or doing anything which would  alter  the  currently
              running kernel.

       -s     Retrieve and display NAT statistics

       -r     Remove  matching  NAT rules rather than add them to the internal
              lists

       -v     Turn verbose mode on.  Displays  information  relating  to  rule
              processing and active rules/table entries.

FILES
       /dev/ipnat

SEE ALSO
       ipnat(5), ipf(8), ipfstat(8)
```

7.2.5 PPP/PPPoE

Two shell commands are supplied with Roadshow which take care of connecting your Amiga to the Internet through the PPP or PPPoE network drivers (these being `ppp-serial.device` and `ppp-ethernet.device`, respectively).

Opening a PPP connection through a modem or ISDN adapter is taken care of by the `ppp_dialer` command.

The `ppp_connector` command is responsible for opening a connection through an ADSL modem.

Each command requires a configuration file, examples of which can be be found in the `S:PPP-Configurations` drawer.

7.2.5.1 ppp_connector

The `ppp_connector` command must be started from shell. It expects the name of the configuration file, and an optional parameter instructing it to keep trying to connect to the PPPoE service if the first connection attempt failed.

The command parameter template looks as follows:

```
NAME/A,RECONNECT/S
```

The individual parameters have the following purpose:

'`NAME/A`'

This must be the name of the configuration file to use. You can use different configuration files for different PPPoE services if you like. But the respective configuration file must be written specifically for the `ppp_connector` command to use: its contents are not entirely compatible with configuration files written for the `ppp_dialer` command.

'`RECONNECT/S`'
: Use this option to make `ppp_connector` retry connecting to the PPPoE service if the first connection attempt failed. Normally, the command will only make one single attempt and exit if it failed.

You can stop the `ppp_connector` program at any time by pressing the *`[Ctrl]+C`* keys, or by using the shell `Break` command.

7.2.5.2 ppp_dialer

The `ppp_dialer` command must be started from shell. It expects the name of the configuration file, and an optional parameter instructing it to keep trying to dial and connect to the PPP service if the first connection attempt failed.

The command parameter template looks as follows:

```
NAME/A,REDIAL/S
```

The individual parameters have the following purpose:

'`NAME/A`'
: This must be the name of the configuration file to use. You can use different configuration files for different PPP dial-up services if you like. But the respective configuration file must be written specifically for the `ppp_dialer` command to use: its contents are not entirely compatible with configuration files written for the `ppp_connector` command.

'`REDIAL/S`'
: Use this option to make `ppp_dialer` retry dialing and connecting to the PPP service if the first connection attempt failed. Normally, the command will only make one single attempt and exit if it failed.

You can stop the `ppp_dialer` program at any time by pressing the *`[Ctrl]+C`* keys, or by using the shell `Break` command.

7.2.5.3 ppp_sample

The `ppp_sample` command can be used to monitor the data throughput of the `ppp-serial.device` or `ppp-ethernet.device` drivers while they are active. This works similarly to the `SampleNetSpeed` program.

You can start the `ppp_sample` program only from shell. The name of the device driver to monitor must be specified. Here is how the command template looks like:

```
NAME/A,UNIT/N,LEFT/N,TOP/N,WIDTH/N,HEIGHT/N
```

The respective options have the following purpose:

'`NAME/A`' This must be the name of the device driver to collect data throughput information for. This parameter is mandatory.

'`UNIT/N`' This gives the unit number of the device driver to collect data throughput information for. If not provided, the unit number 0 will be used.

'`LEFT/N,`'

'`TOP/N,`'

‘WIDTH/N,’
‘HEIGHT/N’
: These parameters define the position and size of the window in which the data throughput information will be displayed. These parameters are all optional.

To stop the `ppp_sample` command, either close its window, hit the *[Ctrl]+C* keys or use the shell `Break` command.

7.2.6 The "TCP:" device

```
NAME
     TCP-Handler -- Access network resources through AmigaDOS

SYNOPSIS
     Open("TCP:[HOST]=<name or address>]/[PORT=<port number>]",...)
     Open("TCP:OBTAIN=<number>",...)

FUNCTION
     TCP-Handler allows AmigaDOS to interface to networking resources,
     such as by connecting to a remote server, exchanging data with it.

     When bsdsocket.library is initialized, it attempts to add a file
     system device by the name of "TCP:". That is, unless there already is
     an assignment, a volume or a file system device of that name.

     The "TCP:" device is something of a pipe which makes simple socket I/O
     (reading and writing) available to the AmigaDOS file system layer. You
     can open connections to other hosts and you can allow other hosts to
     connect to your local machine. How this works out is controlled by the
     parameter with which the device is opened.

TEMPLATE
     H=HOST,P=PORT=S=SERVICE/K,O=OBTAIN/K/N

     (Note that the parameters need to be separated by '/' characters)

OPTIONS
     The individual parameters have the following meanings:

         HOST
             The name or IP address of a host to contact on the network.
             If you specify a host name, you must also specify a port
             number or service name.

         PORT
             The port number or service name to connect to on the remote
             host. If the host parameter is omitted, this means that you
             want to allow other networked hosts to open a connection to
             the local machine on the port specified here.

         OBTAIN
             This will cause a socket to be adopted which has been released
             to the public list. You specify the number, the "TCP:" device
             will try to obtain it via ObtainSocket(). If this parameter is
             used, all others will be ignored.

     The following are equivalent:

         TCP:service=<service name>
         TCP:<service name>

PACKETS
     ACTION_FINDINPUT
     ACTION_FINDOUTPUT
```

```
     ACTION_FINDUPDATE
     ACTION_END
     ACTION_READ
     ACTION_WRITE
     ACTION_WAIT_CHAR
     ACTION_IS_FILESYSTEM
     ACTION_STACK

EXAMPLES
     Print the current time of day (requires that the daytime server
     is enabled in "DEVS:Internet/Servers"):

         1> Type TCP:localhost/daytime
            Thu May 05 09:31:12 2005

SEE ALSO
     bsdsocket.library/ObtainSocket()
```

8 Writing new software for Roadshow

Roadshow uses the AmiTCP V4 API functions, which the majority of Amiga networking software available today supports. In addition to the original AmiTCP V4 API Roadshow also implements a large number of extension functions.

These extension functions are used, for example, by the shell commands that add, configure or monitor the TCP/IP stack operation.

To write new software for Roadshow, and specifically for Roadshow's API extensions, you need the software development kit which is part of the Roadshow software package. You can find everything you should need in the `SDK` drawer.

While the Roadshow software development kit covers the peculiar features of Roadshow, it can also be used in place of the SDK material that shipped for the AmiTCP and Miami TCP/IP stacks.

8.1 Requirements

In order to use the Roadshow software development kit, you will need a 'C' compiler such as SAS/C, or the GNU 'C' compiler that is part of the AmigaOS 4 SDK.

All the source code provided with the Roadshow software development kit was created and tested with the SAS/C and the GNU 'C' compiler.

Of course, an Amiga is required for running the programs, and unless you intend to use a cross-compiler to translate the software, you will need an Amiga system capable of running the respective 'C' compiler.

8.2 The software development kit

The software development kit is stored in the `SDK` drawer that is part of the Roadshow package. Inside the `SDK` drawer you will find the following drawers:

'`sfd`' This contains function interface definitions, required for programs to access the `bsdsocket.library` and `usergroup.library` APIs. These files can be converted to other application specific files, e.g. `"fd"`-files, by the `fd2pragma` program which can be found on Aminet (`http://www.aminet.net/package/dev/misc/fd2pragma.lha`).

'`source_code`'
This directory contains the source code for the Roadshow shell commands, as well as the `tcpdump`, `libpcap` and `wget` commands. Note that the `tcpdump`, `libpcap` and `wget` commands require the use of the GNU 'C' compiler and will not build with the SAS/C compiler.

'`netinclude`'
This directory contains the data structures, constants and API declarations for Roadshow and the TCP/IP stack it is based upon.

'`doc`' Here you will find the documentation for the Roadshow API and the AmiTCP V4 compatible TCP/IP stack API.

'`interfaces`'
This directory contains function interface definitions which are useful mainly for AmigaOS 4, in case you want to rebuild the 'C' interface header files.

8.3 Notes on the source code

The SDK `source_code` drawer contains the source code of the programs that call into the Roadshow `bsdsocket.library` to configure the TCP/IP stack and to query status information.

These programs should compile out of the box, provided you have the necessary SDK header files installed, and a 'C' compiler. I tried to keep all the necessary files together, but it might not be the complete set. Don't worry, though, the code is merely intended to be a demonstration as to how to do what the configuration/query programs are doing.

This software is copyrighted, which means that you may create your own programs using the techniques demonstrated in the source code, and you may reuse parts of it, but you must not take this code as a whole and claim it as your own.

9 Publisher and support

Roadshow is published by

APC-TCP
Andreas Magerl
Postfach 83
D-83234 Übersee
Germany

The web site can be found under `http://www.apc-tcp.de` and a support forum is available under `http://roadshow.apc-tcp.de`.

To directly contact the author, you might want to use the following e-mail address:

obarthel@gmx.net

Index

^

^@ .. ^_ 62

\

\^ 62
\\ 62
\~ 62
\a 62
\b 62
\f 62
\n 62
\r 62
\t 62
\xNN 62
\YYY 62

~

~ 63

7

7WIRE=RTSCTS/K 60

A

AACFC/K 61
AC=ACCESSCONCENTRATOR/K/N 55
ACCM/K 61
ADDRESS/K 37
ALIAS/K/M 37
ALL 54, 60
ARPREQUESTS/K/N 36
ARPTYPE/K/N 35
AUTH 54, 60

B

BPS=SPEED/A/N 61
BUFFERSIZE/K/N 61
BYPASS/K 55

C

CD=CHECKCARRIER/K 61
CONFIGURE/K (possible parameters: DHCP,
AUTO or FASTAUTO) 38
CONNECT 54
CONNECTTIMEOUT/K/N 55
COPYMODE/K (possible
parameters: SLOW or FAST) 36

D

DATAIN 60
DEBUG/K (possible parameters: YES or NO) 36
DEFAULT=DEFAULTGATEWAY/K 43
DESTINATION=DESTINATIONADDR/K 38
DEVICE/K 35, 51, 57
DGRAM=DATAGRAM/S 45
DHCPUNICAST/K 39
DIAL/K 61
DIALTIMEOUT/K/N 61
DIR 49
DNS1ADDRESS/K 51, 57
DNS2ADDRESS/K 51, 57
doc 117
domain 41
DOS/S 45
DOWNGOESOFFLINE/K (possible
parameters: YES or NO) 36
DST=DESTINATION/K 43
DUMMYREMOTEADDRESS/K 53, 59

E

EOF/K 62

F

FILTER/K (possible parameters: OFF, LOCAL,
IPANDARP or EVERYTHING) 36
FILTER=EVERYTHING 37
FILTER=IPANDARP 37
FILTER=LOCAL 37
FILTER=OFF 37
FRAMESIN 54, 60
FRAMESOUT 54, 60

G

GECOS 49
GID/A/N 49

H

HANGUP/K 62
HARDWAREADDRESS/K 37
HEIGHT/N 114
HOSTDST=HOSTDESTINATION/K 43

I

ID 40
ID/K 39
IDLETIMEOUT/K/N 52, 58
IGNOREFCS/K 62
INACTIVE/S 45
INIT/K 62
INTERFACE/K 51, 57
interfaces 117
IPCP 54, 60
ipf -DFa 65, 66
ipf -f S:IPF/ipf.rules -E 65
ipnat -CF 66
ipnat -CF -f S:IPF/ipnat.rules 65
IPREQUESTS/K/N 36
IPTYPE/K/N 35

L

LCP 54, 60
LEASE/K 38
LEASE=1day 39
LEASE=2hours 38
LEASE=300 38
LEASE=300seconds 38
LEASE=30min 38
LEASE=4weeks 39
LEASE=infinite 39
LEFT/N, ... 113
LINK 54, 60
LOCALADDRESS/K 51, 57
LOG/K 54, 60
LOGFILE/K 54, 60
LOGIN/K 54, 60
LQR 54, 60

M

MAXCONFIG/K/N 52, 58
MAXFAIL/K/N 51, 57
MAXHITS/K/N 46
MAXRECONFIGURE/K/N 52, 58
MAXTERM/K/N 51, 58
METRIC/K/N 38
MTU/K/N 38, 52, 58
MULTICAST/K (possible
parameters: YES or NO) 36

N

NAME/A 45, 49, 112, 113
nameserver 41
NAME 40
NETDST=NETDESTINATION/K 43
netinclude 117
NETMASK/K 38
NOREQ/S 46
NULLMODEM/K 63

P

PAPRETRY/K/N 53, 59
PAPTIMEOUT/K/N 53, 59
PASSWORD/K 49, 54, 60
PATH/K/M 45
PEERIDLETIMEOUT/K/N 52, 58
PFC/K 63
POINTTOPOINT/K (possible
parameters: YES or NO) 36
ppp-ethernet0.dns1_address 53
ppp-ethernet0.dns2_address 53
ppp-ethernet0.local_address 52
ppp-ethernet0.peer_address 53
ppp-serial0.dns1_address 59
ppp-serial0.dns2_address 59
ppp-serial0.local_address 59
ppp-serial0.peer_address 59
prefer 41
PRI=PRIORITY/K/N 46
PROGRAM/F 46

R

RAW/K 55
RAW/S 45
READPACKETS/K/N 55
READREQUESTS/K/N 63
RECONNECT/S 113
REDIAL/S 113
REJECTPAP/K 53, 59
REMOTEADDRESS/K 51, 57
REPORTOFFLINE/K (possible
parameters: YES or NO) 36
REQUIRESINITDELAY/K (possible
parameters: YES or NO) 36

S

search 41
SENDID/K 53, 59
SEQPACKET=SEQUENCEDPACKET/S 45
SERVICE/K/N 54
SETENV/K 52, 58
sfd 117
SHARED/K 63
SHELL 49
source_code 117
STACK=STACKSIZE/K/N 46
STATE/K 38
STREAM/S 45

T

TIMEOUT/K/N 52, 58
TOP/N, ... 113

U

UID/A/N 49
UNIT/K/N 35, 51, 57
UNIT/N 113
USERS 40

V

VIA=GATEWAY/K 43
VJHC/K 63

W

WAIT/S 45
WIDTH/N, ... 114
WRITEPACKETS/K/N 55
WRITEREQUESTS/K/N 36, 57