Allied Data
TECHNOLOGIES

Technical Documentation

Project : CopperJet

Author:	Niels Everstijn
Created:	30-03-00 16:44
Version:	0.9
Status:	Draft

This document is subject to change without notice.

History

Author	Date	Version	Status	Description
Niels Everstijn	2000-03-30	0.2	Draft	Initial version
Peter Korsten	2000-04-21	0.3	Draft	Comments from R&D
Niels Everstijn	2000-04-25	0.4	Draft	Roadmap added
Niels Everstijn	2000-05-12	0.5	Draft	Console Commands added
Egbert van de Woude	2000-05-19	0.6	Draft	Supplement
Niels Everstijn	2000-07-21	0.7	Draft	Added Phy commands
Niels Everstijn	2000-08-02	0.8	Draft	Added TCP/IP general information
Niels Everstijn	2000-08-15	0.9	Draft	Added Dual Latency

This document is not officialy released and has to be treated as confidential.

For more information please use our website: www.allieddata.com

This document is not officialy released and has to be treated as confidential.

1 Disclaimer

This manual by ALLIED DATA TECHNOLOGIES B.V. (hereinafter referred to as ALLIED DATA TECHNOLOGIES) is a reflection of the current state of the products described in it. It has been our aim to provide a description which would be sufficiently complete and clear to see to it that our products would be as easy as possible to use. However, this manual may contain technical inaccuracies and typing errors. As a result of rapid developments, we are also obliged to reserve the right to implement technical modifications and developments without prior notice. For this reason, ALLIED DATA TECHNOLOGIES does not warrant the contents of the manual and its permanent applicability.
Neither is ALLIED DATA TECHNOLOGIES liable for possible loss of information or any improper use of information resulting from the consultation of this manual. In particular, ALLIED DATA TECHNOLOGIES is not liable for any direct or indirect damage (including loss of profits and comparable losses) resulting from the use or improper use of this manual, even if ALLIED DATA TECHNOLOGIES or a representative of ALLIED DATA TECHNOLOGIES has been informed that such damage could arise.

Of course, this does not detract from out legal liability for intentionally inflicted damage or damage on the basis of gross negligence. In relation to the information mentioned in this manual, ALLIED DATA TECHNOLOGIES does not warrant that there are no industrial rights of ownership (trademarks, patents, etc.). This also applies to commonly used brand names, company names and product names, but these are subject to the relevant trade mark, patent and registered design laws.

The information is not to be copied, translated, reproduced or transferred or stored on any electronic medium or other machine, neither wholly nor partly, without prior permission in writing from ALLIED DATA TECHNOLOGIES. The sale and use of software is subject to the ALLIED DATA TECHNOLOGIES General Terms of Delivery and Payment as well as its License Terms. Should any term regarding the disclaimer be or become void for legal reasons, this will not affect the other terms
.
© July 2000
Web-Jet, TelTron, QuaTron, TRON-DF , Tele-Talk, Triterm, Trion, Duon and VidiTron
are registered trademarks of

ALLIED DATA TECHNOLOGIES B.V.

IBM is a registered trademark of International Business Machines Corp (IBM).

MNP is a registered trademark of Microcom Inc.

Allied Data Technolgies bv
P.O.Box 788
NL-3200 AS SPIJKENISSE
The netherlands

This document is not officialy released and has to be treated as confidential.

2   General Functionality Overview

2.1   L2TP

Layer 2 Tunnelling Protocol. Extension to PPP that enables ISPs to operate VPNs. Merges features of PPTP (Microsoft) and L2F (Cisco). Tunnelling is achieved by embedding the network protocol within the TCP/IP packets carried by the internet. It is also sometimes called encapsulation.

Allied Data also provides a Layer 2 Tunnelling Protocol (L2TP) client or Access Concentrator (LAC). L2TP has the same primary function as PPTP that is to securely and transparently tunnel PPP data over an unsecured network. L2TP, however, is a far more complex protocol that provides support for advanced security such as IPSec. PPTP is more commonly used in xDSL applications.

L2TP support is limited to dial-in to the L2TP Access Concentrator (LAC). It has been tested on Windows 2000 Release Candidate 2. We have obtained confirmation from Microsoft that L2TP requires IP Security support in the full release of Windows 2000 (build 2195) as confirmed by the problem report "SRH000308600318 - L2TP requires IPSEC post RC2".

L2TP overview:

- L2TP provides tunnelling of PPP over IP.
- Implements the L2TP Access Concentrator (LAC) on ATMOS.
- The L2TP Network Server (LNS) is planned for the future.
- Multiple 'ppp' channels supported in a tunnel.
- Multiple tunnels supported.
- Dial-out and dial-in supported.

2.2   PPTP

Point to point Tunnelling Protocol. Used for creating VPNs across the internet, making sure that the message transmitted is in secure mode. Has been submitted to EITF as standard but currently only available in NT 4.0 and Linux.

The Point-to-Point Tunnelling Protocol (PPTP) provides the ability transfer PPP data through a secure tunnel over a non-secure network such as the Internet. The usefulness is that the physical and logical terminations of the point-to-point link terminate in the unsecured network while the authentication and control terminate in the secure network. This allows, for example, an ISP to provide world wide local dial-in to corporate users. The corporate users dial into the ISP but their data is tunnelled over the Internet to a corporate PPTP network server (PNS). Allied Data has implemented the client portion of PPTP that provides a PPTP Access Concentrator (PAC).

PPP and PPTP have the following limitations: PPTP is unable to handle more than one call per tunnel. The implementation of PPP is known to cause some packet loss at throughputs of over 2 MBps.

PPTP overview:

- PPTP tunnels PPP sessions through an IP network:
    - One end of a tunnel is the PPTP Network Server (PNS), running on a general purpose platform.
    - The other end is the PPTP Access Concentrator (PAC), running on a dial access platform.

- PPTP provides the PAC.
- PNS may be provided in the future.

- Supports:
    - Multiple tunnels
    - Multiple PPP channels per tunnel
    - Dial-out and dial-in

2.3   PPPoE Relay Agent

The name PPPoE indicates that this encapsulation method is used to transport PPP traffic over Ethernet, not ATM. Using this encapsulation allows PPP sessions to be terminated on PCs that are connected to the communications processor by Ethernet. In this case, there may be multiple PPP sessions, each from a PC in the CPE to a PPP aggregator, such as a Redback router, in the CO. These multiple sessions can be to separate end networks (for example Internet and Corporate Network). The PPPoE relay agent recognises when locally originated PPPoE traffic is to be sent to the CO. Such traffic is, without unnecessary processing, forwarded to the correct destination network. This security is useful to prevent, for example, corporate bound data from being exposed to the Internet. The actual ATM encapsulation used in the PPPoE case is actually RFC 1483 because the local user data, though PPP, is encapsulated into Ethernet frames.

2.4   PPPoA (RFC2364)

From a system perspective, the use of PPPoA is similar to IPoA in that user data for transmission is in the form of IP packets. In this case, however, a PPP session is established (using the PPP stack) to the remote NSP. The PPP packets are encapsulated according to RFC 2364 for transmission over an ATM link. On the receive side, the de-encapsulation is performed. The PPP sessionis terminated in the PP and the IP data can be delivered to the end user over, for example, Ethernet.

2.5   DHCP

Allied Data's implementation of the Dynamic Host Control Protocol (DHCP) provides both client and server functions. The client can be used, for example, to obtain a public IP address from an ISP. The DHCP server can be used to configure many local devices with private IP addresses. NAT can then be employed to allow the devices on the private network to send and receive data on the public network by sharing the public IP address.

A DHCP relay uses the facilities of the IP stack to transmit and receive DHCP packets. From a DHCP client's point of view, the relay acts as a de-facto DHCP

server, and this operation is transparent. This is useful where a network administrator wishes to have only one DHCP server across several physical and logical sub-networks.

The relay works by forwarding all broadcast client requests to one or more known DHCP servers. Server replies are then either broadcast or unicast back to the client via the DHCP relay.

Note that DHCP implementation includes code from The Internet Software Consortium.

DHCP requires certain properties of the network interfaces it uses. The interface must be able to broadcast and receive DHCP packets before it has been allocated an IP address.

The ethernet, Forum LAN Emulation (FLANE), and RFC1483 interfaces are all suitable for use with DHCP.

The IP over ATM or Classical IP interfaces (PVC and SVC) are not suitable for use with DHCP. They do not support IP broadcasts and this is an integral requirement of DHCP. No standard exists for DHCP over IPoA. A proprietary solution may be possible but would need to be operating at both ends of an IPoA connection. PPP provides its own mechanism for IP address allocation, obviating the need for DHCP support across this type of interface.

2.6   DNS Relay

DNS Relay is a software module that forwards DNS packets between a DNS resolver and a DNS server. The DNS relay is capable of forwarding query packets from one or more DNS resolvers to exactly one nominated DNS server. DNS responses received from the server are then forwarded back to the DNS resolver that made the original request. Both UDP and TCP traffic are supported.

From the point of view of a DNS resolver, the relay appears to behave exactly as a DNS server. Indeed, the resolver will have its DNS server address configured to match the IP address of the DNS relay. Conversely, from the point of view of the DNS server, the DNS relay appears to be a normal DNS resolver.

The DNS relay does not bind itself to any one specific interface or interface type, but rather will listen for traffic on all available IP interfaces. It relies on the well-known UDP and TCP port number for a DNS server (port number 53) for receiving DNS traffic. This value can be reconfigured, but this is not recommended.

2.7   NAT

The Network Address Translator (NAT) implements Port Address Translation (PAT) and provides Network Address Port Translation (NAPT), also known as IP Masquerading. NAT allows a single "real" IP address on the WAN side to be shared among many devices on the LAN side, each of which have private addresses.

Incoming connections
Normally, NAT is used in a situation where clients on the private network make outgoing connections to servers on the public network. Since the IP addresses on the private network are not visible and cannot be routed to from the public network, it is not possible for a client on the public network to originate a connection directly to a server on the private network. NAT will normally reject any incoming packets that are not in response to a previous outgoing packet.

However, NAT can be configured to allow incoming connections. This is achieved by sending packets destined for a specific port on the one, externally visible IP address to a new machine within the private network. These rules can be added with a configuration application. Packets arriving with different destination port numbers can have their destination IP addresses re-written to different values; for example TCP packets destined for port 80 can be redirected to a web server, and TCP packets sent to port 25 can be sent to a separate mail server.

Application Level Gateway (ALG)
Some application level protocols embed IP address information in the payload of TCP or UDP packets. As NAT itself only modifies information in the packet headers, such protocols require an Application Level Gateway (ALG) in order to operate transparently with NAT.

The NAT process is designed to allow additional Application Level Gateways (ALGs) to be added easily, so that additional or new protocols can be supported. Although all ALGs are an integral part of the NAT process, each ALG is modular, so that only those which are required for a particular application need to be compiled.
Each ALG is required to register with NAT the TCP or UDP port numbers for which it is interested in intercepting packets. It also supplies the addresses of functions which NAT will call for each packet matching that gateway's requirement.

NAT is transparent to the majority of common protocols, including but not limited to:
- DNS queries
- HTTP
- NNTP
- POP3
- SMTP
- SSH (most features)
- Telnet
- TFTP
- Windows drive sharing (most features)

2.8 Sending Dying Gasp message with the Copperjet.

The Copperjet has the ability to detect when the electrical power has been shut off. After such detection of a near-end Loss-of-Power (LPR) condition, the Copperjet will insert emergency priority eoc[1] messages into the ADSL upstream data to implement a "dying gasp" as an LPR indicator.

At least six continuous dying gasp eoc messages will be inserted in the next available ADSL upstream bytes available for eoc. The ATU-C[2] will not send a response to a "dying gasp" message back to the ATU_R[3] (Copperjet).

---

[1] Embedded Operations Channel
[2] ADSL Transceiver Unit, Central office end
[3] ADSL Transceiver Unit, Remote terminal end

At the PCB of the Copperjet, the LPR signal is directly routed to the TI chip TNETD4250. This chip will autonomously start sending the dying gasp signal on the LPR condition, without the software on the Helium being notified. As a consequence, the Copperjet is not able to send, for instance, an SNMP Trap. To enable the Copperjet to do so, the LPR signal should be routed to an interrupt pin of the Helium.

Some additional investigation is required to find out whether the timing of an LPR condition allows an SNMP Trap to be sent in time.

The LPR condition will occur as soon as the power drops for more than 30 mS, which is 3 times the half of a means power cycle. If desired, this can be adjusted by the component values of two resistors on the PCB. In the worst case, the TNETD4250 requires 68 mS to transmit the dying gasp message. The capacitor sizes in the power circuitry take this requirement into account.

2.9   BOOT options

Safe booting from flash:
The system uses flash memory for storing images and configuration information; flash provides a non-volatile method of storing data that allows field updates. The critical period for such a system occurs when the flash memory itself is being updated, as a power failure could result in data corruption and hence an inoperable system.

The flash memory is generally divided into two separate areas:
- Emergency (contains boot code and minimal run-time image)
- Standard (normal run-time image)

The Emergency flash is marked as read-only and cannot be overwritten. The emergency run-time image is used if booting from the normal run-time image fails. Booting from the emergency image allows the user to repair/restore the normal run-time image.

USB boot:
The system uses a file from the PC for booting the system. The file is normally stored in the *WINDOWS\SYSTEM32* directory. Replacing the file upgrades/downgrades the device.

Network boot:(ethernet)
The system will try to find a BootP server during booting. An active BootP server with coorect MAC address settings and a valid boot image must be available in the network.

2.10 Dual Latency

The Copperjet Family has support of two ADSL channels (a0 and a1). Both channels can act independent of eachother. Each channel can be used in a Fastpath or Interleavedpath mode. The actual mode is part of the negotiationphase when a link is build. The CO will force the Copperjet to use a particular mode for one of the two paths.

3   General Console Commands

3.1   event …

Syntax:
```
event help
event n[ext]
event p[revious]
event r[ecent]
event show
event unshow
```

Description:
The command "event show" enables display of background output on this console device; the command "event unshow" disables it. By default, the display of background output is disabled. The command "event recent" (or "event r") displays the most recent background output stored in the memory buffer; "event previous" (or "event p") displays the background output immediately preceding that last displayed; "event next" (or "event n") displays the background output immediately following that last displayed. Up to 24 lines are displayed in each case. For example, after "event r", "event n" will show only new background output that has arrived since the "event r" command: repeated typing of "event n" will let the user keep up to date with new background output (without any repetitions in the output).
The command "event help" displays a summary of the options of the "event" command.

3.2   restart

Syntax:
```
restart
```

Description:
Reboots the ADIOS system.

3.3   uptime

Syntax:
```
uptime
```

Description:
Displays the time for which the system has been up.

3.4   version

Syntax:
```
version
```

Description:
Displays the system type and version.

3.5   <process>, <process> <command>

Syntax:
```
<process> <command>
<process>
home
home <command>
```

This document is not officialy released and has to be treated as confidential.

Description:
In these commands, "<process>" can be any of a list of process names known to the console; see the section on build-time configuration above. The former variant sends the command as a TELL message to the process. The latter variant remembers the process name, and sends subsequent commands as TELL messages to the process, as if they had been preceded by the process name, until the command "home" is issued. The prompt is changed to reflect this; moreover, if a "help" command with no arguments is issued, it is passed to the process as usual, but then information about the "home" command is appended to the process's output by the console.

Example:
```
mymachine> isfs version
ISFS v1.12
mymachine> isfs
mymachine isfs> version
ISFS v1.12
mymachine isfs> help

ISFS commands are:
help            - this text is displayed
ls              - list ISFS files
rm <file>       - remove file from ISFS
cat <file>      - show file contents
version         - displays version number

Use "home" to return to "mymachine>" prompt
mymachine isfs> home
mymachine>
```

When the console is at the prompt of a particular process, the command "home <command>" or "home <process> <command>" may be used to execute a command as if the user had typed "home" followed by "<command>" or "<process> <command>". However, the console will remain at the same process prompt. The command "home <process>" will change the prompt from the current process to a new process "<process>".

Example:
```
mymachine> bridge
mymachine bridge> version
Bridge Version 1.15
mymachine bridge> home version
Modem COPPERJET Version 6.0.0.10 (21 May 1999)
mymachine bridge> home nat version
NAT Version 1.07
mymachine bridge> home edd
mymachine edd> version
EDD Version 1.02
mymachine edd> home
mymachine>
```

See also:
`tell` (in the section on special-purpose commands)

3.6   . (history mechanism)
Syntax:
.

Description:

Repeats the previous console command.

Example:
```
mymachine> isfs version
ISFS v1.12
mymachine> .
ISFS v1.12
```

3.7 "@" commands

Syntax:
```
@@<line>
@ <line>
@<process> <line>
@<process>
```

Description:
Lines beginning with the "@" character are intercepted by the console even when the console device is bound to a file. To bypass this interception and pass a line beginning with "@" to a process, the "@" must be doubled; the line with one "@" removed will be passed on like a normal input line. (At the time of writing, this is most useful when the device is bound to a "slotN" process on a switch; then "@ip" would refer to the ip process on the switch, but "@@ip" would be passed to the slotN process as "@ip" and forwarded by that to the ADIOS console on an expansion card, which will interpret it as referring to the ip process on the expansion card.) If the "@" is followed by a space (or any non-alphanumeric character), the remainder of the line is treated as a console command, as if the device were not bound. The "@<process> <line>" form passes "<line>" to a file (if any) opened for reading by the named process. The "@<process>" form binds the console device to the named process, in the same way as "bind <process>". (Except that the latter, not being an "@" command, will not work if the console device is bound. More generally, "@<process>" does the same as "@ bind <process>".)

Example:
```
mymachine> @ip
```

(The "ip>" prompt does not appear until the "Enter" key is pressed again.).

```
ip> device
# type dev file IP address
device ether ether //nice mtu 1500 192.168.3.55
ip> @console
mymachine>
```

See also:
`bind` (in the section on special-purpose commands)

4 Special-purpose commands

This section lists command that are normally useful only to developers rather than to normal users, or else are retained only for consistency with older versions of the software. They are not described in the output of the "help" command.

4.1 list

Syntax:
```
list
```

Description:
The list command lists the active console devices (referred to as "threads") and files.
For each console device, if it is bound to a file then the list shows which file it is bound to; if background output is enabled on that device (see the *event* command) then the list indicates the fact. For each file, the list shows the name of the process that opened the file and the number of read commands outstanding on the file. If the file is bound to a device then the list shows which device it is bound to; if the file is for foreground output then the list indicates the fact (with the string "FG").

Example:
```
mymachine> list

Threads:
1: ACTIVE, FP 00730520
3: ACTIVE, FP 00719170, Bound 75, events shown
Files:
0: OPEN FP 00718e70, Queue chips, 0 read(s)
1: OPEN FP 00718c30, Queue isfs, 0 read(s)
(some output omitted)
49: OPEN FP 00715af4, Queue ip, 0 read(s), Bound 3, FG
(some output omitted)
75: OPEN FP 00715b38, Queue ip, 1 read(s), Bound 3
(some output omitted)
```

4.2 echo …

Syntax:
```
echo <text>
```

Description:
Echoes the text. (Not a very useful command.)

Example:
```
mymachine> echo hello world
hello world
```

4.3 tell <process> …

Syntax:
```
tell <process> <command>
```

Description:
Sends the command as a TELL message to a specific process. Note that for many processes the "tell" can be omitted; see "<process> <command>" above.

Example:
```
mymachine> tell hswctrl portinfo a1 port type vers flags
```

```
A1 25Mbps 1QUA mast uni30 ilmi netside tx8khz manconfig
```

See also:
```
<process> <command>
```

4.4 exit, exit!

Syntax:
```
exit
exit!
```

Description:
Exits from ADIOS to the boot ROM. Without the exclamation mark the command works only from the serial interface; with the exclamation mark it works from any console device.

4.5 debug

Syntax:
```
debug
```

Description:
Enters the ADIOS debugger. Only works when issued at the serial interface. (Since the ADIOS debugger talks to the serial interface, the "debug" command would be of little use elsewhere.)

4.6 crlf, nocrlf

Syntax:
```
crlf
nocrlf
```

Description:
Controls whether line-feed characters written to this console device are output as carriage-return/ line-feed pairs (crlf) or just as single line-feed characters (nocrlf).

4.7 bind <process>, unbind

Syntax:
```
bind <process>
unbind
```

Description:
The former command binds this console device to the specified process — that is, binds this device to the file, if any, opened for read by that process, and binds every file opened by the process to this device. The latter command unbinds this console device — that is, undoes the above bindings.

Example:
```
mymachine> bind ip
ip> @ unbind
mymachine>
```

See also:
"@" commands

5 Commands for the "chips" process

5.1 cpu

Syntax:
```
cpu
```

Description:
Displays the recent CPU utilization as a percentage. This is a fairly crude measurement: the ADIOS kernel measures the time that the CPU spends in the idle loop over successive three-second intervals, and the "cpu" command uses this measurement from the most recent complete three-second interval.

5.2 debug

Syntax:
```
debug
```

Description:
Enters the ADIOS debugger.

5.3 exit

Syntax:
```
Exit
```

Description:
Exits from ADIOS to the boot ROM.

5.4 help

Syntax:
```
help
help <command>
```

Description:
The "help" command lists all chips commands; "help <command>" displays more detailed help on the specified command. This is available only if the pre-processor symbol CHIPSHELP is defined.

5.5 info

Syntax:
```
info
```

Description:
Displays system type and version number, and the MSNL and MAC addresses.

5.6 mem

Syntax:
```
mem
```

Description:

Displays a summary of how much memory is used by each ADIOS process (distinguishing between heap and thread stacks, along with some other minor categories), along with the amount of free heap memory and the size of the largest single free block. It should be noted that (with the kernel behavior at the time of writing) the memory for a process's main stack is not attributed to the process; instead, all process stacks are attributed to the "idle" process.

5.7   rb, rw, wb, ww

Syntax:
```
rb <addr>
rw <addr>
wb <addr> <val>
ww <addr> <val>
```

Description:
Reads the byte or word at a specified address, or writes a specified value to the byte or word. Addresses and values are specified in hexadecimal, with an optional "0x" prefix.

Example:
```
rw 1c4b54
word at 0x001C4B54 contains 0x0000337E
rb 1c4b55
byte at 0x001C4B55 contains 0x33
wb 1c4b56 0x20
value 0x20 written to byte at 0x001C4B56
rw 1c4b54
word at 0x001C4B54 contains 0x0020337E
ww 0x1c4b54 14c44
value 0x00014C44 written to word at 0x001C4B54
>
```

5.8   rlogin

Syntax:
```
rlogin [-a] <MSNL-addr> [<port>]
rlogin -i <IP-addr>
rlogin -s <MSNL-addr> [<port>]
```

Description:
Available only in debug builds.
Performs an RLOGIN or telnet connection to a given MSNL or IP address. By default, or with the -a flag, uses raw AAL5 (a standard RLOGIN); the default port is 0.0.0.130.
With the -i flag, uses telnet (to the standard telnet port). Telnet commands are not understood, and may be displayed as strange bytes.
With the -s flag, uses SSCOP; the default port is 0.0.0.142. (This is of little use, since standard ADIOS systems do not now listen at that SSCOP port.)

5.9   steal

Syntax:
```
steal memory use <handle> <amount>
steal memory release <handle>
steal file use <handle> <device>
steal file release <handle>
steal cpu use <percentage>
steal cpu release
steal status [memory] [file] [cpu]
```

Description:
Uses up heap memory, file handles, or CPU cycles. <handle> is a number from 0 to 19, used to identify the resource for a later "steal … release" command.
This command is intended to help test system behavior when resources are limited, and is available only if the pre-processor symbol CHIPS_STEAL is defined.

5.10 tell

Syntax:
```
tell <process> <command>
```

Description:
Sends the command as a TELL message to a specific process. (The same as the console "tell" command.)

5.11 time

Syntax:
```
time
```

Description:
Displays the uptime (the time for which the system has been up); also displays the current date and time, which will be valid if the system has booted from a boot server but not (on standard ADIOS systems) if it has booted from flash.

5.12 top

Syntax:
```
top [<n>]
```

Description:
Sends an MSNL topology request and displays the replies; this explores the local ATM network, so far as MSNL is enabled, and shows its topology — which port on which machine is connected to which port on which other machine. <n> is the number of receive requests submitted in parallel; it defaults to 32, but a larger value may be needed on a large network in order to avoid missing replies.

6 PPP Console commands

Console commands should be prefixed with `ppp` in order to direct them to the **ppp** process.

6.1 *<channel> clear*

Syntax:
`<channel> clear`

Description:
Clear all aspects of this channel back to their default settings. If there is an active connection it is torn down.

6.2 *<channel> disable*

Syntax:
`<channel> disable`

Description:
Clear the enable flag for a PPP channel. This is the default setting. Disabling does not remove other configured information about this channel.
In the PPP state machine, this sets the PPP link to 'closed'. If it is already closed, there is no effect. Configuration saving saves this information. By default all channels are disabled.

See also:
`<channel> enable`

6.3 *<channel> discard*

Syntax:
`<channel> discard [<size>]`

Description:
Discard is a PPP LCP packet type, which is like the Echo packet type but does not generate a return. This can be used for more careful tests of data transfer on the link, for instance at sizes near the negotiated MRU. This command sends an LCP Discard packet, of the specified size. If no size is given, a minimal sized packet is sent. Arrival of a Discard packet is logged locally as a level 2 event. The link must be up and operational in order to do the discard test.

See also:
`<channel> echo`

6.4 *<channel> echo*

Syntax:
`<channel> echo [<size>]`

Description:
Echo is an LCP packet, which is used to test an established PPP link. It solicits a ping-like reply from the far end. This command sends an LCP Echo packet, of the specified size. If no size is given, a minimal sized packet is sent. If a size greater than the remote Maximum Receive Unit size is specified, the value is reduced to the remote MRU before sending. The command waits for 1 second for a reply packet to arrive, and prints whether the reply arrived. If a reply arrives subsequent to this, it is logged as a level 2 event. The link must be up and operational in order to do the echo test.

15-08-00

See also: the `discard` test.

6.5 *<channel> echo every*

Syntax:
`<channel> echo every <seconds>`

Description:
Echo is an LCP packet, which is used to test an established PPP link. It solicits a ping-like reply from the far end.
This command sets a channel to confirm the continued presence of an open PPP connection by sending an LCP echo every few seconds, and requiring an echo reply. The number of seconds between echo requests is specified as a parameter.
If 0 is specified, the function is disabled. Use the `info all` command to read the current state on a channel. Configuration saving saves this information. By default the function is disabled.

See also:
`echo (manually initiated LCP echo)`
`info all (show current state)`

6.6 *<channel> enable*

Syntax:
`<channel> enable`

Description:
Set the enable flag for a PPP channel. By default this is disabled.
In the PPP state machine, this flag sets the PPP link to 'open'. If it is already open, there is no effect. Configuration saving saves this information. By default all channels are disabled.

See also:
`disable (reverse the effect)`
`status`

6.7 *<channel> event*

Syntax:
`<channel> event [<n>]`

Description:
Read or set the overall trace output level.
Configuration saving does not save this value. The default event level is 1.
Event levels are:
1 only very serious errors reported (default)
2 definite protocol errors or very significant events reported
3 links going up/down are reported
4 every packet and significant state change is reported
5 every packet sent/received is disassembled, and hex dumped

6.8 *<channel> hdlc*

Syntax:
`<channel> hdlc [1|0]`

Description:
If 1, use an HDLC header on the front of transmitted packets and require one on received ones. This consists of two bytes, FF-03, and assists in interoperability with some other (non-standard) implementations. If 0, disable this. Call with no argument to find the current setting. The default value is 0 (disabled). Configuration saving saves this information.
If not set, and a packet is received with an HDLC header, the channel goes into a 'learned HDLC' mode and sends packets with the HDLC header. Thus, interoperation with HDLC-using equipment should not normally require any configuration. Learning occurs in this direction only. Setting `hdlc` to 0 clears this learned state. Configuration saving does not save the learned state.

6.9   *<channel> info*

Syntax:
```
<channel> info [all]
```

Description:
Provide information about the current settings of this channel. This includes all configured state, and also current protocol information. Specifying 'all' prints out more information. `info` and `status` are synonyms.

6.10 *<channel> interface*

Syntax:
```
<channel> interface <n>
```

Description:
Logically associate the specified channel with the specified interface. Interface 1 is always the router port. It should be used for any PPP channel over which IPCP communication with the local system's IP router is desired. Other interfaces can be created for bridging. A single PPP channel can only be associated with a single interface, or a single tunnel. Use `info` to find the current setting. Calling with n=0 removes any association. This is the default state. Configuration saving saves this information.

See also:
```
<channel> info
<channel> tunnel <n>
```

6.11 *<channel> lcpmaxconfigure*

Syntax:
```
<channel> lcpmaxconfigure [<n>]
```

Description:
Set the Max-Configure parameter for LCP, as described in section 4.6 of RFC1661. This is the maximum number of Configure Requests that will be sent without reply, before assuming that the peer is unable to respond. Call with no argument to find the current setting.
The default value is 10. Configuration saving saves this information.

6.12 *<channel> lcpmaxfailure*

Syntax:
```
<channel> lcpmaxfailure [<n>]
```

Description:

Set the Max-Failure parameter for LCP, as described in section 4.6 of RFC1661. This is the maximum number of consecutive Configure Naks that will be sent before assuming that parameter negotiation is not converging. Call with no argument to find the current setting. The default value is 5. Configuration saving saves this information.

6.13 *<channel> lcpmaxterminate*

Syntax:
```
<channel> lcpmaxterminate [<n>]
```

Description:
Set the Max-Terminate parameter for LCP, as described in section 4.6 of RFC1661. This is the maximum number of Terminate Requests that will be sent without reply, before assuming that the peer is unable to respond. Call with no argument to find the current setting. The default value is 2. Configuration saving saves this information.

6.14 *<channel> llc*

Syntax:
```
<channel> llc [1|0]
```

Description:
If 1, use an LLC header on the front of transmitted packets and require one on received ones. This consists of four bytes, FE-FE-03-CF, and is required for PPP Over AAL5 (RFC 2364 p4) when using LLC encapsulated PPP. If 0, disable this. Call with no argument to find the current setting. The default value is 0 (disabled). Configuration saving saves this information. If not set, and a packet is received with an LLC header, the channel goes into a 'learned LLC' mode and sends packets with the LLC header. Thus, interoperation with LLC-using equipment should not normally require any configuration. Learning occurs in this direction only. Setting `hdlc` to 0 clears this learned state. Configuration saving does not save the learned state.

6.15 *<channel> pvc*

Syntax:
```
<channel> pvc [[<port>] <vpi>] <vci> [ip|mac] [listen]
<channel> pvc none
```

Description:
Attach an ATM PVC to the given PPP channel. The port can be specified (only for a multi-port device), and the VPI (default is 0), and the VCI. The allowable range of port, VPI, VCI depends on the atm driver. Normal limits are 0 only for port, 0 only for VPI, 1..1023 for VCI. If a single argument `none` is supplied, any current connection is torn down. This is equivalent to `svc none` on the channel. In the PPP state machine, providing a link of this form causes the link to be 'up'. Note that `enable` must also be used, to allow the link to become operational. The `ip` or `mac` indicates which form of data is transported over the connection: one of IP data (controlled by the IPCP protocol), or MAC data (for BCP). If neither is provided, `ip` is assumed. If the channel is not linked to an interface, and the channel is for IP data, the channel is linked to interface 1. If the channel is not linked to an interface, and the channel is for MAC data, the channel is linked to interface 2. Providing a PVC setting unsets any SVC setting. See the `svc` command. It is possible for a PVC to become 'down' in the PPP state machine even though the PVC is still there, for instance due to an authentication failure. If in this state, an incoming packet will cause the PPP state machine to go 'up'.
If `listen` is specified then this is the server end of a PVC. It will not send out PPP Configure Requests until it first receives a packet over the PVC. When a connection is torn down it goes returns to this state. Use the `info` command to read this information.

Configuration saving saves this information. By default a channel has no connection information.

Example:

`ppp 3 pvc 3 32` set channel 3 to be (VPI=3, VCI=32)
`ppp 4 pvc` read PVC settings for channel 4
`ppp 5 pvc 0` remove any PVC settings from channel 5

6.16 *<channel>* qos

Syntax:
`<channel> qos [cbr|ubr] [pcr <pcr-tx> [<pcr-rx>]]`

Description:
Specify that the VC for a PPP channel should be Constant Bit Rate or Unspecified Bit Rate, and (optionally for UBR) give a Peak Cell Rate for the connection. If two values are specified then they are the transmit and receive PCRs respectively.
If called while not attached to a VC then the settings are saved for use when a VC is created. If the channel is already attached to a VC then it is closed, and re-opened with the new values. If it cannot be reopened, it remains closed. Configuration saving saves this information. By default channels are established UBR.

Example:
`ppp 3 qos cbr pcr 10000 set channel 3 to be CBR limited at 10000 cells/sec`

6.17 *<channel>* remoteip

Syntax:
`<channel> remoteip [<ipaddress>]`

Description:
If a PPP link is established using IPCP, this call causes the channel to provide the given IP address to the remote end of the connection. PPP will refuse to complete the connection if the other end will not accept this. This is normally used for channels on which the remote party dials in, to allocate the IP address to that remote party.
Call with no argument to find the current setting. Call with 0.0.0.0 to remove any setting. This is the default state. Configuration saving saves this information.

See also:
`interface <n> localip`

6.18 *<channel>* svc

Syntax:
`<channel> svc listen [ip|mac]`
`<channel> svc addr <addr> [ip|mac]`
`<channel> svc none`

Description:
Specify that the VC for a PPP channel should be an SVC (i.e. created by signaling). This can either be by listening for an incoming call, or by making an outgoing call to a specified ATM address. The outgoing call or listen occurs immediately. If the call fails it will be retried after a few seconds. In the PPP state machine, providing a connection of this form causes the channel to be 'up' or 'down'. Note that `enable` must also be used, to allow the link to become operational. Outgoing and incoming UNI signalling calls are identified by a BLLI value that identifies PPP. (Aside: A BLLI of length 3 bytes is used, hex values 6B, 78. C0.) If the channel

is already attached to an SVC or PVC then it is closed, and re-opened with the new settings. If it cannot, it remains closed. If a single argument `none` is supplied, any current connection is torn down. This is equivalent to `pvc none` on the channel. The `ip` or `mac` indicates which form of data is transported over the connection: one of IP data (controlled by the IPCP protocol), or MAC data (for BCP). If neither is provided, `ip` is assumed. Providing an SVC setting unsets any PVC setting. See the `pvc` command. Configuration saving saves this information. By default a channel has no connection information.

Example:
```
ppp 3 svc
47.00.83.01.03.00.00.00.00.00.00.00.00.00.20.2b.00.03.0b.00
ppp 4 svc listen (listen for incoming call)
ppp 7 svc none (tear down connection, remove setting)
```

6.19 <channel> theylogin
Syntax:
```
<channel> theylogin pap|chap|none
```

Description:
This command describes how we require the far end to log in on this channel. Requiring the other end to log in most frequently happens when they dial us (rather than the other way round), so this is likely to be one of several channels which are set using `svc listen`. Because of this, exact names and passwords are not attached to individual channels but are matched to particular users, as defined using the `user` command. This command specifies that when using this channel, the user must log on using the specified protocol, and that they must provide any name/password combination which has been defined for that protocol, using the `user` command.
To remove this information on a channel, call `theylogin` with a single argument of `none`. Configuration saving saves this information. By default no login is required.

6.20 <channel> tunnel <n> <tunnel protocol> <dial direction>
Syntax:
```
<channel> tunnel <n> <tunnel protocol> <dial direction>
```

Description:
Logically associate the specified channel with the specified tunnel. A single PPP channel can only be associated with a single interface, or a single tunnel. Use `info` to find the current setting. Calling with n=0 removes any association. This is the default state. Configuration saving saves this information.

The possible tunnel protocols are: `pptp` and `l2tp`.
The dial directions may be: `in` or `out` for dial-in or dial-out respectively.

Example:
```
ppp 3 tunnel 1 pptp out
```

See also:
```
<channel> info
<channel> interface <n>
```

6.21 <channel> welogin
Syntax:
```
<channel> welogin <name> <password> [pap|chap]
```

```
<channel> welogin none
```

Description:
This command describes how we should log in to the far end when a connection is established. A name and password are supplied, and whether these should be used with the PAP or CHAP authentication protocol. CHAP is the default. To remove this information on a channel, call `welogin` with a single argument of `none`. If `chap` is specified, we will also log in using `pap` if the other end prefers this. If `pap` is specified we will only log in using `pap`. Configuration saving saves this information. By default no login is performed.

6.22 bcp
Syntax:
```
bcp stp|nostp
```

Description:
This command describes parameters for BCP, the Bridge Control Protocol, which is used to transport MAC (Ethernet) packets over the PPP link. See the protocol conformance section of this spec for BCP option settings, which are not controllable. If `stp` is specified, the Spanning Tree Protocol is in use by the Bridges, to control bridge loops. In this case STP frames should be carried over any links using BCP. If `nostp` is specified, STP frames should not be carried. Configuration saving saves this information. By default STP is not supported.

6.23 interface <n> localip
Syntax:
```
interface <n> localip <address>
```

Description:
This command describes parameters for IPCP, the IP Control Protocol, when providing the server end of an IPCP connection. The server knows its own IP address (and may allocate an IP address to the remote end). This command tells the PPP process, for a particular interface, the local IP address to be associated with the local end. For interface 1, this should be the same IP address as possessed by the device `ppp_device` in the IP stack. See the IP dial-in server console example, at the start of this section. If PPP channels are now associated with this interface, remote users can dial in to those channels and will be connected to the IP stack. They can be allocated IP addresses, see the command

<channel> remoteip.

Call with 0.0.0.0 to remove any IP address setting. This is the default state.
Configuration saving saves this information.

See also:
```
<channel> remoteip
```

6.24 interface <n> stats
Syntax:
```
interface <n> stats
```

Description:
The interface is regarded by the operating system as an Ethernet-like device, which can be attached to the bridge or router, like other Ethernet devices in ADIOS. It also provides an ifEntry to SNMP providing basic information about traffic through the interface. This command shows the basic information about byte and packet traffic through the interface, in SNMP terms.

6.25 user

Syntax:
```
user add <name> [pwd <passwd> [pap|chap]]
user [<name>]
user delete <name>|all
```

Description:
This command stores information about a particular login name/password combination. This is referred to as a 'user', regardless of whether it represents an individual. When `user` is called on its own, information about all existing users is listed. When `user <name>` is called with no further arguments, details of that user alone are printed. Passwords are not shown. Use `user delete` to delete an individual user by name, or to delete all users.
Use `user add <name>` to create a new user or update an existing one. The password is stored, and the authentication protocol which must be used for this user. If a user is deleted or changed, existing sessions are not affected. Configuration saving saves this information.

6.26 version

Syntax:
```
version
```

Description:
Provide the version number for the source of the **ppp** process.

7 Spanning Tree Console Commands

Commands are directed to the spanning tree "process" by sending commands to the **bridge** process and preceding such commands with the keyword `spanning`. Depending on the console interface provided, it may be necessary to precede `spanning` with `bridge`, for

Example:
`bridge spanning status` Display the status of the spanning tree.

7.1 *disable*

Syntax:
`disable`

Description:
Disables the spanning tree process. When spanning tree operation is disabled, the bridge operates in transparent mode and all bridge ports are set to the forwarding state. The `status` command reports the enabled state of the spanning tree process. Configuration saving saves this information. By default, spanning tree operation is enabled.

Example:
`spanning disable` Disables spanning tree operation

See also:
`enable`
`status`

7.2 *enable*

Syntax:
`enable`

Description:
Enables the spanning tree process. When spanning tree operation is enabled, the spanning tree process controls the state of the bridge's ports. The `status` command reports the enabled state of the spanning tree process. Configuration saving saves this information. By default, spanning tree operation is enabled.

Example:
`spanning enable` Enables spanning tree operation

See also:
`disable`
`status`

7.3 *forwarddelay*

Syntax:
`forwarddelay [<time>]`

Description:
Reads or sets the time in seconds, in which the bridge remains in the listening or learning states, and is used when the bridge is or is attempting to become the root bridge. The forward delay time may be any value between 4 and 30 but it is also constrained by the maximum age and hello times. The forward delay time may also by changed by SNMP command. The maxage, hellotime and forwarddelay times are constrained as follows:

$2 \times$ (forwarddelay - 1) $\geq$ maxage
maxage $\geq 2 \times$ (hellotime + 1)

Configuration saving saves this information. By default the forward delay time is set to 15 seconds.

Example:
`forwarddelay 10` Sets the forwarding delay to 10 seconds.
`forwarddelay` Reports the current forward delay time.

See also:
`hellotime`
`maxage`

7.4 hellotime

Syntax:
`hellotime [<time>]`

Description:
Reads or sets the time in seconds, after which the spanning tree process sends notification of topology changes to the root bridge, and is used when the bridge is or is attempting to become the root bridge. The hello time may be any value between 1 and 10 and is also constrained by the forwarddelay and maxage times. The hello time may also by changed by SNMP command. Configuration saving saves this information. By default the hello time is set to 2 seconds.

Example:
`hellotime 5` Sets the hello time to 5 seconds

See also:
`forwarddelay`
`maxage`

7.5 info

Syntax:
`info`

Description:
Displays the version number of the spanning tree implementation.

Example:
`info`

See also:
`version`

7.6 maxage

Syntax:
`maxage [<time>]`

Description:
Reads or sets the maximum age of received spanning tree protocol information before it is discarded, and is used when the bridge is or is attempting to become the root bridge. The maxage time may be any value between 6 and 40 and is also constrained by the

forwarddelay and hellotime times. The maxage time may also by changed by SNMP command. Configuration saving saves this information. By default the maxage time is set to 20 seconds.

Example:
`maxage 6` Sets the maxage time to 6 seconds.

See also:
`forwarddelay`
`hellotime`

7.7 port <number>

The port commands, described in subsequent sections, control the configuration of the bridge's ports so far as the operation of the spanning tree protocol is concerned. Ports are numbered from 1. Every port on the bridge may be specified by typing `all` instead of a port number.

7.8 port <number> disable

Syntax:
`port <number> disable`

Description:
Allows a port to be disabled. The state of a port may also be changed by SNMP command. A port, which is enabled, will not take part in the operation of the spanning tree protocol. Configuration saving saves this information. By default ports are enabled.

Example:
`port 2 disable` Disables port 2 on the bridge.

See also:
`port <number> enable`
`status`

7.9 port <number> enable

Syntax:
`port <number> enable`

Description:
Allows a port to be enabled. The state of a port may also be changed by SNMP command. A port, which is enabled, will take part in the operation of the spanning tree protocol. If enabled, the physical port may be "enabled" or "disabled" as demanded by the operation of the protocol.
Configuration saving saves this information. By default ports are enabled.

Example:
`port 1 enable` Enables port 1 on the bridge.

See also:
`port <number> enable`
`status`

7.10 port <number> pathcost

Syntax:
```
port <number> pathcost [<cost>]
```

Description:
Reads or sets the cost of using this port. The cost may be any number between 1 and 65535. The cost of the port is used when deciding which is the best path to the root bridge. The cost of a port may also be changed by SNMP command.
Configuration saving saves this information. By default a cost of 10 is assigned to a port.

Example:
```
port 2 pathcost
``` Displays the path cost for port 2 on the bridge

**See also:**
port <number> priority

### 7.11  port <number> priority

**Syntax:**
```
port <number> priority [<portpriority>]
```

Description:
Reads or sets the priority of the port. The priority may be any value between 0 and 255. The priority is used in conjunction with the pathcost to determine the best root to the root bridge. The higher the priority number, the less significant, in protocol terms, the port. The port priority may also be changed by SNMP command.
Configuration saving saves this information. By default a port has a priority of 128.

Example:
```
port 1 priority
``` Displays the priority for port 1 on the bridge

**See also:**
```
priority
port <number> pathcost
```

7.12 priority

Syntax:
```
priority [<bridgepriority>]
```

Description:
Reads or sets the priority of the bridge. The priority may be any value in the range 0 to 65535. The higher the priority number, the less significant, in protocol terms, the bridge. Where two bridges have the same priority, their MAC address is compared and the smaller MAC address is treated as more significant. The priority of the bridge may be changed by SNMP command.
Configuration saving saves this information. By default the bridge is assigned a priority of 32768.

Example:
```
priority 4000
``` Sets the bridge priority to 4000.

**See also:**
port <number> priority

### 7.13  status

**Syntax:**
```
status
```

Description:
Reports the status of the spanning tree. If spanning tree operation is disabled, a message is printed to that effect and no other information is displayed. When spanning tree operation is enabled, the following information is displayed:
- The identifier of the bridge.
- The identifier of the root bridge.
- The root port for this bridge.
- The root path cost: how far the bridge is from the root.
- The various spanning tree time values as defined by the current root bridge:
- The maximum age of spanning tree information before it is discarded: max age time.
- The amount of time between configuration protocol packets: hello time.
- The amount of time delay when ports are changing state: forward delay time.
For each port:
- The identifier of the designated bridge
- The identifier of the designated port for the designated bridge
- The identifier of the designated root bridge

Example:
```
status
```

See also:

7.14 version

Syntax:
```
version
```

Description:
Displays the version number of the spanning tree implementation.

Example:
```
version
```

8   Transparent Bridge Console Commands

Console commands should be prefixed with `bridge` in order to direct them to the **bridge** process.

8.1   *device add*

Syntax:
```
device add <device>
```

Description:
This command adds a device to the bridge configuration. Attempts to add the bridge itself or an existing device to the bridge are rejected. Attempts to add devices, which don't support the Cyan interface, are rejected. There is a limit on the number of devices that can be attached to the bridge. If a device is successfully added to the bridge, it will only become active after the configuration is saved and the system is rebooted. If the device being added is from a process which supports multiple devices, the /DEVICE attribute must be specified as part of the device name. The table below shows devices, which may be attached to the bridge, although not all systems may support all devices.

Device	Remarks	Source
lec1	Forum LAN emulation	alecjade
edd	Ethernet driver	bun_ethernet
r1483	RFC1483 protocol (PVC)	rfc1483
ppp	Point-to-Point protocol	pp

Configuration saving saves this information. See the section implementation constraints for details of which devices are added by default.

Example:
```
device add edd
device add ppp/DEVICE=2
```

See also:
device delete, device list

8.2   *device delete*

Syntax:
```
device delete <device>
```

Description:
This command deletes a device from the bridge configuration. The changes will only take place after the configuration is saved and the system is rebooted. The syntax of the device name is the same as that for the `device add` command.
Configuration saving saves this information.

Example:
```
device delete r1483
```

See also:
device add, device list

8.3   *device list*

Syntax:

```
device list
```

Description:
This command lists all the devices that are currently attached to the bridge. It does not show the stored configuration (which can be seen with the `config print` command).

Example:
```
device list
```

See also:
```
device add, device delete
```

8.4   ethertype
Syntax:
```
ethertype [<port> any|ip|pppoe]
```

Description:
This command enables filtering of Ethernet packets according to the ETHER_TYPE field in the header. Only packets of the type specified using this command will be **sent** on the port specified; packets of all types will always be **receive**d. By default, all bridge ports are set to "any", which means that the type of the packet will never be checked. The meaning of the other options is as follows:

Option Permitted ETHER_TYPE values
"ip" 0x0800   – IP
0x0806   – ARP
"pppoe" 0x8863, 0x8864   – PPP Over Ethernet (RFC 2516)

The port is specified as an integer, as displayed by the `device list` command. When using this command in the `initbridge` configuration file, ports are numbered in the order in which the `device add` commands are given, starting from 1. If no arguments are given, the current settings for each port are displayed.

Example:
```
ethertype 2 any
```

See also:

8.5   filter
Syntax:
```
filter
```

Description:
This command shows the current contents of the bridge's filter table. The MAC entries for each device are shown in turn together with the time that the bridge last saw the MAC address. The command also shows the current filter aging time, in seconds, and the number of creation failures since the system was started. Creation failures occur when there is no room left in the filter table for a new entry.

Example:
```
filter
```

See also:
```
filterage
```

8.6 filterage

Syntax:
```
filterage [<age>]
```

Description:
This command sets, or displays if no arguments are given, the filter table aging time. The aging time is the time after which MAC addresses are removed from the filter table when there has been no activity. The time is specified in seconds and may be any integer value in the range 10…100,000 seconds. This value may also be changed through SNMP. Changing the value of filterage has immediate effect. Configuration saving saves this information. By default the filter ageing time is set to 300 seconds.

Example:
```
filterage
```

See also:
```
filter
```

8.7 flush

Syntax:
```
flush [<port>]
```

Description:
This command allows the MAC entries for a specified port, or all ports, to be removed from the filter table. The port number for a device may be determined using the `device list` or `status` commands. If the port number is omitted, all entries for all ports are removed from the filter table.

Example:
```
flush
```

See also:
filter, device list, status

8.8 info

Syntax:
```
info
```

Description:
This command displays build information about the **bridge** process. The `version` command is a synonym.

Example:
```
info
```

See also:
```
version
```

8.9 interface

Syntax:
```
interface [sub-command]
```

Description:

This command accesses the ethernet support library sub-commands for the bridge itself, not for the devices, which are attached to it. The ethernet support commands are documented in the "ADIOS Ethernet Support Library" specification.

Example:
interface stats

See also:

8.10 portfilter

Syntax:
portfilter [<source port> all|<destination ports>]

Description:
The portfilter command allows control over the bridge's forwarding and broadcasting behavior. By default, when a multicast or an unknown packet is received on a port (referred to above as the source port), it will be forwarded to all other bridge ports (referred to above as the destination ports). Each bridge port may have its behavior modified separately. The first example below configures the bridge so that packets arriving on port 2 will only be forwarded to ports 3, 4 and 5, and packets arriving on port 3 will only be forwarded to port 1. All other ports retain their default behavior. Note that this command does not force packets arriving on the source port to be sent to all specified destination ports.
The bridge retains its learning behavior, so unicast packets, once the bridge knows their destination, will still only be sent to one port. Note also that the bridge itself (for example when attached to the IP router) will always forward to all ports, and will always be forwarded to by all ports. Calling this command with the argument "all", as shown in the second example can restore the default behavior. The ports are specified as integers, as displayed by the device list command. When using this command in the initbridge configuration file, ports are numbered in the order in which the device add commands are given, starting from 1.
If no arguments are given, the current settings for each port are displayed.

Example 1:
portfilter 2 3 4 5
portfilter 3 1

Example 2:
portfilter 2 all
portfilter 3 all

See also:

8.11 spanning

Syntax:
spanning [sub-command]

Description:
The spanning tree commands are only available if it has been compiled in to the bridge. The spanning tree commands are documented in the "ADIOS Spanning Tree" specification.

8.12 status

Syntax:
Status

Description:

This command shows the status of the bridge and its ports. The status information for a port includes the SNMP type information about time exceeded packets, packets discarded, etc. It also includes the broadcast history of the port over the last five seconds and the high water mark of packets queued on the bridge for this device.

Example:
status

See also:

8.13 version

Syntax:
version

Description:
This command displays build information about the **bridge** process. The info command is a synonym.

Example:
version

See also:

9 RFC1483 Console Commands

Console command should be prefixed with rfc1483 in order to direct them to the rfc1483 process.

9.1 event

Syntax:
```
event [<level>]
```

Description:
This command sets or displays the generation of messages on unusual events. Set this to:

2 report receive LLC/SNAP header errors
3 level 2 reports, and also transient resource failures
Bridge PDUs link start/stop

Configuration saving saves this information. By default, the event level is set 0.

Example:
r1483 event 2

See also:

9.2 help

Syntax:
```
help
```

Description:
This command displays the commands understood by the RFC1483 process.

Example:
r1483 help

See also:

9.3 info

Syntax:
```
info
```

Description:
This command displays build information about the RFC1483 process.

Example:
r1483 info

See also:

9.4 interface

Syntax:
```
interface [sub-command]
```

Description:
This command access the Ethernet interface sub-commands (see [2]).

Example:
```
r1483 interface stats
```

See also:

9.5   pvc

Syntax:
```
pvc [<channel>|none] [port]
```

Description:
This command sets or displays the PVC used for communications. When setting the PVC, the configuration must be saved and the system restarted before the change takes effect. The argument may be `none` to indicate no PVC configured, or a value in the range 1.maxVCI. maxVCI is typically 1023 but is fixed by system configuration.
On BUN (Broadband Unified Network) based systems it is also possible to set, or display, the port with which the PVC is associated. Configuration saving saves this information. By default, there is no PVC.

Example:
r1483 pvc 12

See also:

9.6   status

Syntax:
status

Description:
This command displays the status of the RFC1483 process. At present, the status consists of whether the process is active, that is has a valid PVC, or is inactive, that is has no PVC.

Example:
r1483 status

See also:

10 TCP/IP Console Commands

10.1 *abort*

Syntax:
```
abort <assoc>
```

Description:
Aborts an IP association; <assoc> is the number of the association as shown by the "files" command. Currently (ADIOS IP version 1.29) this seems to be unreliable on UDP associations and can cause a crash (possibly because of lax error-handling by the application that opened the file); it is reliable on TCP associations.
The "abort" command is "hidden", not shown by "ip help"; it is probably useful, if at all, for debugging and troubleshooting.

Example:
```
mymachine> ip abort 3
```

See also:
```
files
```

10.2 *arp*

Syntax:
```
arp add <i/f> <IP address> <MAC address>
arp delete <i/f> <IP address>
arp flush
arp [list]
arp help [all|<cmd>]
```

Description:
Allows display and manipulation of the ARP table: the list of IP addresses and corresponding MAC addresses obtained by ARP (see 4.3.1) on Ethernet-like interfaces. Normally there is no need to add entries to the table with "arp add", since they should be discovered by the ARP protocol. Displaying the table with "arp list" (or just "arp") is sometimes useful, and deleting an entry with "arp delete", or the whole table with "arp flush", can sometimes speed up recovery from temporary problems if something unusual has happened. Entries added with "arp add" do not time out like those discovered by use of the ARP protocol, but they are deleted by "arp flush" and will not survive a restart (they are not saved by configuration saving).
Note that the ARP table is used only for destinations on directly connected Ethernet-like networks, not for those reached through routers (although the ARP table may be used to discover the MAC address of the router).

Example:
```
mymachine> ip arp add ether 192.168.50.1 8:0:20:19:9A:D9
mymachine> ip arp
arp add flane 192.168.2.63 00:20:2b:e0:03:87 # 8m58s
arp add flane 192.168.2.108 00:20:2b:03:0a:72 # 7m02s
arp add flane 192.168.2.109 00:20:2b:03:08:b1 # 2m24s
arp add flane 192.168.2.156 00:20:2b:03:09:c4 # 1m01s
arp add ether 192.168.50.1 08:00:20:19:9a:d9 # forever
arp add ether 192.168.50.57 00:20:af:2e:fa:3c # 3m25s
mymachine> ip arp delete flane 192.168.2.109
mymachine> ip arp list
arp add flane 192.168.2.63 00:20:2b:e0:03:87 # 8m46s
arp add flane 192.168.2.108 00:20:2b:03:0a:72 # 6m50s
arp add flane 192.168.2.156 00:20:2b:03:09:c4 # 49s
```

```
arp add ether 192.168.50.1 08:00:20:19:9a:d9 # forever
arp add ether 192.168.50.57 00:20:af:2e:fa:3c # 3m13s
mymachine> ip arp flush
mymachine> ip arp
# flane ARP table is empty
# ether ARP table is empty
mymachine> ip arp
arp add flane 192.168.2.108 00:20:2b:03:0a:72 # 10m58s
# ether ARP table is empty
```

(The last example shows that the MAC address for 192.168.2.108 has been automatically added again, having been discovered by means of the ARP protocol.)

10.3 arprouting

Syntax:
```
arprouting [on]|off [<i/f>]
```

Description:
The "arprouting" command was intended to control whether a router would perform proxy ARP on an Ethernet-like interface; that is, reply with its own MAC address to an ARP request for any IP address that it would route to. However, it is not supported and is believed currently (ADIOS IP version 1.29) not to work correctly; the command is "hidden", not shown by "ip help".

10.4 autoloop

Syntax:
```
autoloop [on|off]
```

Description:
Displays or sets the "autoloop" setting. This has effect only when a loopback device is configured (see 4.3.3): in that case, it controls whether datagrams addressed to the machine's own IP addresses (and not just the loopback addresses 127.*.*.*) will be looped back. Configuration saving saves this information. By default autoloop is disabled.
The "autoloop" command is "hidden", not shown by "ip help".

Example:
```
mymachine> ip autoloop
autoloop off
mymachine> ip device
# type dev file IP address
device ether ether //nice mtu 1500 192.168.2.1
device loop loop - mtu 2048 127.0.0.1
mymachine> ip ping 127.0.0.1
ip: ping - reply received from 127.0.0.1
mymachine> ip ping 192.168.2.1
ip: ping - transmit error: Host is down (rc=62)
mymachine> ip autoloop on
mymachine> ip ping 192.168.2.1
ip: ping - reply received from 192.168.2.1
```

10.5 config

Syntax:
```
config [save]
```

Description:
Displays the IP configuration (not including the "snmp" configuration), or saves it in flash memory. The functionality of the "config" command is also accessible in the standard way through the config process (e.g. "config print ip"), if that process is present. However, when accessed through the config process, the "snmp" configuration *is* included.

Example:
```
mymachine> ip config
device add ether ether //nice mtu 1500 192.168.2.1
device add vlane ether //lane mtu 1500 192.168.55.1
subnet add vlane.home . 192.168.55.0 ff:ff:ff:00
subnet add ether.home . 192.168.2.0 ff:ff:ff:00
rip send ether 2
rip send vlane 2
rip accept ether 1 2
rip accept vlane 1 2
autoloop on
route add default 0.0.0.0 192.168.2.7 00:00:00:00 2 # MAN
relay ether ether
relay ether vlane
relay vlane vlane
ipatm lifetime 60
# IP host table:
# Port table:
router 520/UDP
snmp 161/UDP
tftp 69/UDP
telnet 23/TCP
mymachine> ip config save
Updating flash filing system ...
done
ip: configuration saved
```

See also:
```
snmp
```

Commands used for setting configuration displayed and saved by "config":
autoloop, device, ipatm, iphostname, portname, relay, rip, route, subnet

10.6 device
Syntax:
```
device
device add <i/f> <type> [<file>] [mtu <size>] [<IP address>]
device delete <i/f>
device flush
```

Description:
Displays the interfaces that IP is configured to use, or adds an interface to the configuration, or deletes an interface, or all interfaces, from the configuration.
Currently (ADIOS IP version 1.29), however, the commands to change the configuration do not take effect immediately (except when the "device add" command is run at start-up from t he initialization file). It is necessary to save the configuration (e.g. with "ip config save") and restart the system (e.g. with "ip restart") before they take effect. "device" will display both the current interfaces and those that have been configured but are not yet in effect. (Other commands apply only to the devices in effect, rather than to those configured; when adding a device, for example, one may need to issue the "device add" command, then the "config

save" and reboot, then issue any other configuration commands that depend on the existence of the device, and then "config save" again.)

"<i/f>" is an arbitrary label for the interface, which is used in referring to it in subsequent commands. (It is often chosen to be the same as "<type>", though this is perhaps slightly confusing.)

"<type>" specifies the class of interface: Ethernet-like, IP-over-ATM, or loopback, as described in section 4.3. For an Ethernet-like or IP-over-ATM interface,
"<file>" specifies the file name that will be opened to access the underlying device (which must support the Emerald interface for an Ethernet-like interface, and the Blue interface, at least, for an IP-over-ATM interface). For a loopback interface, "<file>" is not used, and can just be specified as "-" or omitted altogether.

Example:
```
mymachine> ip device
# type dev file IP address
device ether ether //nice mtu 1500 192.168.2.1
device vlane ether //lane mtu 1500 192.168.55.1
mymachine> ip device add loop loop 127.0.0.1
Change will have no effect until after config save and restart.
mymachine> ip device delete vlane
Change will have no effect until after config save and restart.
mymachine> ip device
# type dev file IP address
device ether ether //nice mtu 1500 192.168.2.1
device vlane ether //lane mtu 1500 192.168.55.1 # DELETED
device loop loop - mtu 2048 127.0.0.1 # ADDED
Additions/deletions will have no effect until after config save and
restart.
```

See also:
```
enable
subnet
```

10.7 disable
Syntax:
```
disable [<i/f>]
```

Description:
Disables all interfaces, or just a specified interface.

Example:
```
mymachine> ip disable vlane
mymachine> ip device
# type dev file IP address
device ether ether //nice mtu 1500 192.168.2.1
device vlane ether //lane mtu 1500 192.168.55.1 # DISABLED
```

See also:
```
device
enable
```

10.8 enable
Syntax:
```
enable [<i/f> [mtu <size>] [<IP address>]]
```

Description:
Enables all interfaces, or just a specified interface. Can also be used to set the MTU and IP address on an interface when enabling it (or change them on an interface that is already enabled); see the "device" command for details on these.
Configuration saving saves the MTU and IP addresses, but not the disabled/enabled state.

Example:
```
mymachine> ip enable vlane 192.168.56.3
ip/vlane: IP address 192.168.56.3
mymachine> ip device
# type dev file IP address
device ether ether //nice mtu 1500 192.168.2.1
device vlane ether //lane mtu 1500 192.168.56.3
```

See also:
```
device
disable
subnet
```

10.9 errors

Syntax:
```
errors
```

Description:
Turns on tracing of various unusual events; equivalent to "trace errors".
The "errors" command is "hidden", not shown by "ip help".

Example:
```
mymachine> ip errors
ip: currently tracing errors
```

See also:
```
noerrors
trace
```

10.10 etherfiles

Syntax:
```
etherfiles
```

Description:
Lists the file names for the underlying devices for all Ethernet-like interfaces, and displays the number of packets (usually none) that are queued for transmission awaiting an ARP reply.
The "etherfiles" command is "hidden", not shown by "ip help".

Example:
```
mymachine> ip etherfiles
ether: //nice, 1 queued for tx
vlane: //lane
atm: (no ethernet device)
```

See also:
```
device
```

10.11 files

Syntax:
```
files [full]
files <assoc>
```

Description:
Lists the files (associations) that other applications (or, internally, RIP) have opened on "//ip". More detailed information on an association can be displayed by specifying the association number, or on all associations by specifying "full".

The information for each association may include an interface name ("ether" or "vlane" in the example below). This can be either the interface last used to send a packet on the association or, for a new association, the interface that is expected to be used for packets to the remote host. This interface can change over the lifetime of an association; in particular, for a UDP association not bound to a specific remote host it may change each time a packet is sent to a different destination. (In other cases it will normally change only as a result of routing changes.)

The "files" command is "hidden", not shown by "ip help".

Example:
```
mymachine> ip files
1: rw+ ether 192.168.2.1 TCP port telnet (23) Established to
192.168.2.2
port 1071 1 rx requests
2: rw+ ether <noaddr> UDP port snmp (161) 3 rx requests
3: rw+ <unset> <noaddr> UDP port tftp (69) 4 rx requests
4: rw+ <unset> <noaddr> UDP port router (520) 2 rx requests
5: w vlane <noaddr> UDP port router (520)
6: rw+ <unset> <noaddr> UDP port 2050 4 rx requests
7: rw+ <unset> <noaddr> UDP port 2051 4 rx requests
8: rw+ <unset> <noaddr> UDP port 2052 4 rx requests
9: rw+ <unset> <noaddr> UDP port 2053 4 rx requests
mymachine> ip files 3
3: rw+ <unset> <noaddr> UDP port tftp (69) 4 rx requests
//ip/TYPE=UDP/LPORT=69/TIMEOUT_CONX=1000/TIMEOUT_LISTEN=0/TIMEOUT_IDL
E=0/
RETRY_CONX=2/TOS=routine/DELAY=normal/THROUGHPUT=normal/RELIABILITY=n
ormal/BU
FFERRX=off/BUFFER_TXSIZE=-1/BUFFER_RXSIZE=-
1/FRAGMENT=on/TTL=60/OPTIONS=off/C
HECKSUM=on/TIMEOUT_USER=540000
```

10.12 flush

Syntax:
```
flush <assoc>
```

Description:
Given an association number (see "files" command) that corresponds to a TCP association, this does a TCP "push" (see RFC 793), which, roughly speaking, causes the data sent s o far to be delivered as quickly as possible to the recipient, without waiting to be buffered with subsequent data.

The "flush" command is "hidden", not shown by "ip help"; it is of little or no use.

See also:
```
files
```

10.13 get

Syntax:
```
get <file>
```

Description:
Reads and executes commands from a file. The commands in the file are in the same format as those documented in this chapter, with no "ip" prefix. They can contain comments, introduced by the "#" character.
The "get" command is "hidden", not shown by "ip help".

Example:
```
mymachine> ip get //isfs/cmdfile
```

10.14 help

Syntax:
```
help
help <cmd>
help all
```

Description:
Displays a summary of available commands, more detailed information on a particular command, or more detailed information on all commands.
(As described in section 7.1, some commands are "hidden" and are not displayed by "help" or "help all"; help is still available on these using the "help <cmd>" form if one knows the name of the command.)

Example:
```
mymachine> ip help

Commands are:

? arp config device
disable enable help ipatm
norelay ping relay restart
rip route routes snmp
stats subnet uptime version

Use "ip help all" or "ip help <command>" for syntax

mymachine> ip help arp
arp syntax:
arp <cmd> - execute arp subcommand
arp help - list subcommands available
```

10.15 ipatm abort

Syntax:
```
ipatm abort <n>
```

Description:
Closes an IP-over-ATM SVC; the number <n> is as displayed by "ipatm files". If there is still traffic being sent to the destination concerned, IP will soon open a new SVC to the destination.

Example:
```
mymachine> ip ipatm abort 14
```

See also:
```
ipatm files
```

10.16 ipatm arp
Syntax:
```
ipatm arp [list]
```

Description:
Lists the cached mappings from IP addresses to ATM addresses; only relevant when using IP-over-ATM with SVCs. (The "list" parameter is optional and makes no difference to the behavior.)

Example:
```
mymachine> ip ipatm arp
192.168.5.72   47.00.83.10.a2.b1.00.00.00.00.00.00.00.20.2b.01.00.07.00
192.168.5.33   47.00.83.10.a4.00.00.00.00.00.00.00.00.20.2b.01.00.19.00
192.168.5.111  47.00.83.10.e2.00.00.00.20.2b.01.01.a8.00.20.2b.01.01.a8.00
```

10.17 ipatm arpserver
Syntax:
```
ipatm arpserver [<i/f> [<ATM address>|here]]
```

Description:
Displays or sets the ATMARP server used for an interface, which must be an IP-over-ATM interface using SVCs. The interface name is optional when displaying: if omitted, the ATMARP servers for all such interfaces are listed. (Since currently there can only be one such interface, this behaviour is present only for possible consistency with future versions.) The parameter "here" causes no ATMARP server to be used; only the local ATMARP cache will be consulted when setting up an SVC. This will normally be used when this machine is the ATMARP server for the local network.
Configuration saving saves this information.

Example:
```
mymachine> ip ipatm arpserver
ipatm arpserver atm here
mymachine> ip ipatm arpserver atm 47.0.83.10.a2.0.0.0.0.0.0.0.0.0.20.2b.4.3.8.0
mymachine> ip ipatm arpserver atm
ipatm arpserver atm 47.00.83.10.a2.00.00.00.00.00.00.00.00.20.2b.04.03.08.00.
```

10.18 ipatm files
Syntax:
```
ipatm files
```

Description:
Lists the IP-over-ATM connections, listens, and slots for available connections.

Example:
```
mymachine> ip ipatm files
i/f atm 0 transmissions queued, 6 free connections, 4 listeners
0: on atm Connected to 192.168.220.48, 2 rx buffers idle 0ms
1: on atm Listening, 1 rx buffers (in use)
2: on atm Listening, 1 rx buffers (in use)
3: on atm Listening, 1 rx buffers (in use)
4: on atm Listening, 1 rx buffers (in use)
5: on atm Idle, 0 rx buffers
```

```
6: on atm Idle, 0 rx buffers
7: on atm Idle, 0 rx buffers
8: on atm Idle, 0 rx buffers
9: on atm Idle, 0 rx buffers
10: on atm Idle, 0 rx buffers
```

10.19 ipatm help

Syntax:
```
ipatm help [<cmd>|all]
```

Description:
Displays help on "ipatm" subcommands.

Example:
```
mymachine> ip ipatm help

Commands are:
? abort arp arpserver files help lifetime pvc

Use "ip ipatm help all" or "ip ipatm help <command>" for syntax

mymachine> ip ipatm help arp
arp syntax:
ipatm arp [list] - list ARP cache entries
```

10.20 ipatm lifetime

Syntax:
```
ipatm lifetime <secs>
```

Description:
Displays or sets idle time-out for IP-over-ATM SVCs: if there is no traffic on an SVC for this period, then it will be disconnected. (It might be disconnected before this period in order to make room for new connections.) There is no way to disable the time-out, but "ip ipatm lifetime 999999" will have much the same effect. Configuration saving saves this information. The default lifetime is 60 seconds.

Example:
```
mymachine> ip ipatm lifetime

Idle lifetime for connections: 1m
mymachine> ip ipatm lifetime 90
Idle lifetime for connections: 1m30s
```

10.21 ipatm pvc

Syntax:
```
ipatm pvc
ipatm pvc add <i/f> <vci>/[<IP address>][/<pcr>] [<port>]
ipatm pvc delete <vci> [<port>]
ipatm pvc flush
```

Description:
Lists configured PVCs for use by IP-over-ATM; configures another; deletes one; or deletes all.
"<i/f>" is the name of an interface configured for IP-over-ATM using PVCs.
"<vci>" is the VCI to use for the PVC. The range of possible VCIs depends on the system.

15-08-00

"<IP address>" is the IP address of the machine at the other end of the PVC. If it is not specified, ADIOS TCP/IP will use Inverse ATMARP (RFC 1577) to determine the IP address; if it is specified, then Inverse ATMARP will not be used.
"<pcr>" is the peak cell rate, in cells per second. The default is 60000. (If neithe r IP address nor PCR is specified, the "/" after the VCI can be omitted.)
"<port>" is the port name: it must be specified if the machine is a switch, and not otherwise.
Configuration saving saves this information.

Example:
```
myswitch> ip ipatm pvc add atm 60 a3
myswitch> ip ipatm pvc add atm 61//50000 b1
myswitch> ip ipatm pvc add atm 62/192.168.4.32 b1
myswitch> ip ipatm pvc
ipatm pvc atm 60//60000 A3
ipatm pvc atm 61//50000 B1
ipatm pvc atm 62/192.168.4.32/60000 B1
```

10.22 iphostname

Syntax:
```
iphostname add <IP address> <name>
iphostname flush
iphostname list
iphostname help [all|<cmd>]
```

Description:
Sets up a mapping between an IP address and a symbolic name; deletes all such mappings; lists the mappings; or displays help on the "iphostname" command.
The symbolic names can be used in most IP commands where an IP address is required, and as values of the attributes LHOST and RHOST (section 6.1). They are also displayed and returned as attribute values in place of numerical addresses, when a suitable mapping exists.
The Damson interface (5.4.2) allows other processes to query the mapping.
The "iphostname" command is "hidden", not shown by "ip help".
Configuration saving saves this information.

10.23 noerrors

Syntax:
```
noerrors
```

Description:
Undoes the effect of the "errors" command; equivalent to "untrace errors".
The "noerrors" command is "hidden", not shown by "ip help".

Example:
```
mymachine> ip noerrors
ip: currently tracing nothing
```

See also:
```
errors
untrace
```

10.24 norelay

Syntax:
```
norelay [all | <i/f> [<i/f>] [forward]]
```

Description:
Turns off forwarding between interfaces; see the "relay" command for more details.
The command "norelay" with no parameters is equivalent to "norelay all": it turns off all forwarding. Configuration saving saves this information.

Example:
```
mymachine> ip relay
relay ether ether
relay ether vlane
relay vlane vlane
mymachine> ip norelay ether vlane forward
relay ether ether
relay vlane ether forward
relay vlane vlane
```

See also:
relay

10.25 ping

Syntax:
```
ping <IP address> [<ttl> [<size>]]
```

Description:
Sends an ICMP Echo message to the specified IP address.
"<ttl>" (default 30) is the TTL (time-to-live) to use. A crude "traceroute" functionality can be obtained by repeating the "ping" command with increasing TTL values, starting with 1.
"<size>" (default 56) is the data size of the Echo message. This does not include the IP header (20 bytes) and the ICMP header (8 bytes).
ADIOS TCP/IP waits 10 seconds for a reply to the message; if none arrives, it reports the lack of a reply (and returns the TELL message, or redisplays the prompt). Any reply arriving after this time-out will be reported as a background message. (Whereas a reply arriving before the time-out expires is, of course, reported in the foreground.)
A reply is an ICMP Echo Reply message, or an ICMP error message reporting destination unreachable, time exceeded, or (as should never happen) a parameter problem. ICMP redirect and source quench messages are reported, but ADIOS TCP/IP continues to wait for a final reply or time-out.

Example:
```
mymachine> ip ping 192.168.4.13 1
ip: ping - 192.168.1.9 reports pkt #5834 to 192.168.4.13: time-to-
live exceeded
mymachine> ip ping 192.168.4.13 2
ip: ping - reply received from 192.168.4.13
mymachine> ip ping 192.168.77.77
ip: ping - no reply received
```

10.26 portname

Syntax:
```
portname add <name> <number>[/<protocol>]
portname flush
portname list
portname read <file>
portname help [all|<cmd>]
```

Description:

Sets up a mapping between a UDP or TCP port and a symbolic name; deletes all such mappings; lists the mappings; reads the mappings from a file; or displays help on the "portname" command.

The symbolic names can be used as values of the attributes LPORT and RPORT provided the protocol type (UDP or TCP) is appropriate. They are also displayed in place of port numbers, when a suitable mapping exists. The Damson interface allows other processes to query the mapping.

"<protocol>" should be either "UDP" or "TCP"; it can be omitted, but that is not very useful. For "portname read", the file is in the sam e format as //isfs/services (section 5.3), which is the same as the output from "portname list". The "portname" command is "hidden", not shown by "ip help". Configuration saving saves this information.

Example:
```
mymachine> ip portname flush
mymachine> ip portname add someport 105/tcp
mymachine> ip portname list
someport 105/TCP
mymachine> ip portname read //isfs/services
mymachine> ip portname list
router 520/UDP
snmp 161/UDP
tftp 69/UDP
telnet 23/TCP
someport 105/TCP
```

10.27 protocols

Syntax:
```
protocols
```

Description:
Displays information on the protocols supported by ADIOS TCP/IP. The output will always be the same for a given version of ADIOS TCP/IP. The "protocols" command is "hidden", not shown by "ip help". (Currently, ADIOS IP version 1.29, there is a fault in the output: it claims falsely that UDP can reassemble fragments.)

Example:
```
mymachine> ip protocols
ICMP - IP ID 1, CL protocol, can't reassemble fragments
TCP - IP ID 6, CO protocol, can't reassemble fragments
UDP - IP ID 17, CL protocol, can reassemble fragments
```

10.28 relay

Syntax:
```
relay
relay all | <i/f> [<i/f>] [forward]
```

Description:
Displays or sets what forwarding ADIOS TCP/IP will do between interfaces. The combinations of setting forwarding can be a bit confusing; they behave as follows:
Command: Enables forwarding: relay all from every interface to every non-loopback interface relay if1 from if1 to every non-loopback interface, and from every interface to if1 relay if1 forward from if1 to every non-loopback interface relay if1 if2 from if1 to if2 and from if2 to if1 relay if1 if2 forward from if1 to if2 (Don't confuse the "forward" keyword, which indicates one-way relaying, with the term "forwarding"!)
To disable forwarding, use the "norelay" command. Configuration saving saves this information. By default all forwarding is disabled.

Example:
```
mymachine> ip relay
No relaying is being performed
mymachine> ip relay ether vlane forward
relay ether vlane forward
mymachine> ip relay ether forward
relay ether ether
relay ether vlane forward
mymachine> ip relay ether vlane
relay ether ether
relay ether vlane
mymachine> ip relay all
relay ether ether
relay ether vlane
relay vlane vlane
```

See also:
```
norelay
```

10.29 restart

Syntax:
```
restart
```

Description:
Reboots the system.

Example:
```
mymachine> ip restart
```

10.30 rip accept

Syntax:
```
rip accept [all|<i/f>] [none|<version>*]
```

Description:
Controls for which version or versions of RIP (RIP version 1, RFC 1058, or RIP version 2, RFC 1723) ADIOS TCP/IP will accept incoming information on each interface. Configuration saving saves this information. By default both RIP versions are accepted on all interfaces ("rip accept all 1 2 ").

Example:
```
mymachine> ip rip accept all 1 2
mymachine> ip rip accept ether 2
mymachine> ip rip allowed
rip send ether none
rip send vlane none
rip accept ether 2
rip accept vlane 1 2
See also:
rip allowed
rip send
```

10.31 rip allowed

Syntax:

```
rip allowed
```

Description:
Displays the RIP versions that will be accepted and sent on each interface.

Example:
```
mymachine> ip rip allowed
rip send ether 2
rip send vlane 2
rip accept ether 1 2
rip accept vlane 1 2
```

See also:
```
rip accept
rip send
```

10.32 rip boot

Syntax:
```
rip boot
```

Description:
Broadcasts a request for RIP information from other machines. ADIOS TCP/IP does this automatically when it first starts up, and the routing information should be kept up to date by regular broadcasts from the other machines, so this command is normally of little use.

Example:
```
mymachine> ip rip boot
```

10.33 rip help

Syntax:
```
rip help [<cmd>|all]
```

Description:
Displays help on "rip" subcommands.

Example:
```
mymachine> ip rip help

Commands are:
? accept allowed boot help hostroutes killrelay poison relay relays
rxstatus send trigger

Use "ip rip help all" or "ip rip help <command>" for syntax

mymachine> ip rip help boot
boot syntax:
rip boot - broadcast RIP request for routes
```

10.34 rip hostroutes

Syntax:
```
rip hostroutes [off]
```

Description:

Sets or clears the "hostroutes" flag; ADIOS TCP/IP will accept RIP routes to individual hosts only if this flag is on. If the flag is off, then RIP version 1 routes that appear to be to individual hosts will be treated as if they were to the network containing the host; RIP version 2 routes to individual hosts will be ignored. (The reason for this difference is that RIP version 1 does not allow specification of subnet masks; a RIP version 1 route that appears to be to an individual host might in fact be to a subnet, and treating it as a route to the whole network may be the best way to make use of the information.)
To see the state of the flag without changing it, the "config" command must be used. Configuration saving saves this information. By default the "hostroutes" flag is off.

Example:
```
mymachine> ip rip hostroutes off
```

See also:
config

10.35 rip killrelay

Syntax:
```
rip killrelay <relay>
```

Description:
Deletes a RIP relay. See "rip relay" for information on RIP relays.

See also:
rip relay

10.36 rip poison

Syntax:
```
rip poison [off]
```

Description:
Sets or clears the "poisoned reverse" flag. If this flag is on, ADIOS TCP/IP performs "poisoned reverse" as defined in RFC 1058; see that RFC for discussion of when this is a good thing. To see the state of the flag without changing it, the "config" command must be used. Configuration saving saves this information. By default the "poisoned reverse" flag is off.

Example:
```
mymachine> ip rip poison
```

10.37 rip relay

Syntax:
```
rip relay <RIP version> <name> [<i/f> [<timeout>]]
```

Description:
Configures a RIP relay. RIP relays were designed as a means of using RIP on a non-broadcast medium (currently, only IP-over-ATM); on such an interface, ADIOS TCP/IP will send RIP information individually to each configured RIP relay, instead of broadcasting it. However, the RIP relay support has not been recently tested and is not believed to be reliable; furthermore, configuration saving does not save the RIP relay configuration. On a non-broadcast medium, therefore, it is preferable to use static (manually configured) routes.

See also:
rip killrelay

```
rip relays
```

10.38 rip relays

Syntax:
rip relays

Description:
Displays the configured RIP relays. See "rip relay" for information on RIP relays.

See also:
rip relay

10.39 rip rxstatus

Syntax:
rip rxstatus

Description:
Displays the status of the RIP packet reception mechanism. This command is of little or no use except for debugging.

Example:
```
mymachine> ip rip rxstatus
RIP has 2 reading threads and 1 worker
The worker is waiting for something to do
The readers have filled 0/6 buffers and have 4 available
Maximum work queue size was 0
Receiver 0 has 1 buffer and is not waiting for the worker
Receiver 1 has 1 buffer and is not waiting for the worker
```

10.40 rip send

Syntax:
rip send [all|<i/f>] [none|<version>*]

Description:
Controls which version or versions of RIP (RIP version 1, RFC 1058, or RIP version 2, RFC 1723) ADIOS TCP/IP will use to broadcast routing information on each interface. If both versions are specified, routing information is broadcast in duplicate, once using each version. Specifying "all" affects all interfaces except the loopback interface (if any). Configuration saving saves this information.
By default RIP version 2 only is used on all non-loopback interfaces ("rip send all 2").

Example:
```
mymachine> ip rip send all 2
mymachine> ip rip send ether 1
mymachine> ip rip allowed
rip send ether 1
rip send vlane 2
rip accept ether 1 2
rip accept vlane 1 2
See also:
rip accept
rip allowed
```

10.41 rip trigger

Syntax:
```
rip trigger
```

Description:
Triggers broadcast of routing information. Normally the routing information is broadcast every 30 seconds; "rip trigger" causes it to be sent almost immediately rather than waiting for the next time it is due. This command is normally of little use.

Example:
```
mymachine>  ip rip trigger
```

10.42 route

Syntax:
```
route
route add <name> <dest> <relay> [<mask> [<cost> [<timeout>]]]
route delete <name>
route flush
```

Description:
Lists routes; adds or deletes a static route; or deletes all routes.
"<name>" is an arbitrary name specified to "route add" that can be used to delete the route using "route delete".
"<dest>" is the IP address of the network being routed to (only those bits of "<dest>" corresponding to bits set in "<mask>" are relevant).
"<relay>" is the IP address of the next-hop gateway for the route.
"<mask>" (default ff:ff:ff:00) is the subnet mask of the network being routed to, specified as four hexadecimal numbers separated by colons. For example, 0:0:0:0 is a default route (matches everything without a more specific route), ff:ff:ff:0 would match a Class C network, and ff:ff:ff:ff is a route to a single host. (Note: the default is not always sensible; in particular, if "<dest>" is 0.0.0.0 then it would be better for the mask to default to 0:0:0:0. This may change in future versions.)
"<cost>" (default 1) is the number of hops counted as the cost of the route, which may affect the choice of route when the route is competing with routes acquired from RIP. (But note that using = a mixture of RIP and static routing is not advised.)
"<timeout>" (default 0, meaning that the route does not time out) is the number of seconds that the route will remain in the routing table.
Note that the routing table does not contain routes to the directly connected networks, without going through a gateway. ADIOS TCP/IP routes packets to such destinations by using the information in the device and subnet tables instead.
The "route" command (with no parameters) displays the routing table. It adds a comment to each route with the following information:

• How the route was obtained; one of
    MAN — configured by the "route" command
    RIP — obtained from RIP
    ICMP — obtained from an ICMP redirect message
    SNMP — configured by SNMP network management;
• The time-out, if the route is not permanent;
• The original time-out, if the route is not permanent;
• The name of the interface (if known) that will be used for the route

• An asterisk ("*") if the route was added recently and RIP has not yet processed the change (the asterisk should disappear within 30 seconds, when RIP next considers broadcasting routing information).

Configuration saving saves this information. (Only the routes configured by the "route" command are saved or displayed by "config".)

Example:
```
mymachine> ip route add default 0.0.0.0 192.168.2.3 0:0:0:0
mymachine> ip route add testnet1 192.168.101.0 192.168.2.34
mymachine> ip route add testnet2 192.168.102.0 192.168.2.34 ff:ff:ff:0 1 60
mymachine> ip route
route add testnet2 192.168.102.0 192.168.2.34 ff:ff:ff:00 1 # MAN 58s/1m via ether *
route add testnet1 192.168.101.0 192.168.2.34 ff:ff:ff:00 1 # MAN via ether
route add default 0.0.0.0 192.168.2.3 00:00:00:00 1 # MAN via ether
```

See also:
device
subnet

10.43 routeflush

Syntax:
```
routeflush [<i/f>] [all]
```

Description:
Removes routes from the route table. If "<i/f>" is specified, only routes through the named interface are removed. If "all" is not specified, only host routes (those with a mask of ff:ff:ff:ff) are removed.
The "routeflush" command is "hidden", not shown by "ip help". Configuration saving saves this information.

Example:
```
mymachine> ip routeflush ether all
mymachine> ip routeflush
```

See also:
route

10.44 routes

Syntax:
```
routes
```

Description:
Lists routes. (The same as "route", with no parameters.)

Example:
```
mymachine> ip routes
route add testnet1 192.168.101.0 192.168.2.34 ff:ff:ff:00 1 # MAN via ether
route add default 0.0.0.0 192.168.2.3 00:00:00:00 1 # MAN via ether
```

See also:
route

10.45 snmp

Syntax:
```
snmp access [read|write|delete|flush] <parameters>
snmp config [save]
snmp help [<cmd>|all]
snmp trap [add|delete|flush|list] <parameters>
```

Description:
Manages the list of SNMP community names (also used as passwords by other applications, such as telnet) and the list of SNMP trap destinations. See chapter 9 for the interface to this information. The syntax of the commands is documented in the ADIOS SNMP Functional Specification [2].

Note that in standard ADIOS systems the console is configured to allow the commands to be accessed by typing just "snmp …" instead of "ip snmp …" at the command line.

10.46 stats

Syntax:
```
stats arp|icmp|ip|tcp|udp [reset]
stats help [<cmd>|all]
```

Description:
Displays or clears a subset of IP statistics.

Example:
```
mymachine> ip stats udp
ip: UDP receptions delivered to users: 0
ip: UDP receptions with no users: 170
ip: Otherwise discarded UDP receptions: 0
ip: Transmitted UDP packets: 35
mymachine> ip stats udp reset
mymachine> ip stats udp
ip: UDP receptions delivered to users: 0
ip: UDP receptions with no users: 0
ip: Otherwise discarded UDP receptions: 0
ip: Transmitted UDP packets: 0
```

10.47 subnet

Syntax:
```
subnet
subnet add <name> <i/f> <IP address> <mask>
subnet delete <name>
subnet flush
```

Description:
Lists defined subnets; defines a subnet; deletes a subnet definition; or deletes all subnet definitions. "<name>" is a label, that can be specified by "subnet add" and later used by "subnet delete" to delete the subnet. "<i/f>" is not used, but is present for historical reasons and must be specified as either "." or a valid interface name. "<IP address>" is the IP address of the subnet being defined (only those bits of "<dest>" corresponding to bits set in "<mask>" are relevant).
"<mask>" is the subnet mask of the subnet being defined, specified as four hexadecimal numbers separated by colons. A subnet is defined automatically for each interface, with a name formed by appending ".home" to the device name. The only significant use for the "subnet" command is to change the masks for these automatic subnets, if the default masks (see "device" command) are not correct. (Subnet definitions for other subnets *can* also be useful in conjunction with RIP version 1, which does not communicate subnet masks, but this is not very common.)

Configuration saving saves this information.

Example:
```
mymachine> ip device
```

This document is not officialy released and has to be treated as confidential.

```
# type dev file IP address
device ether ether //nice mtu 1500 192.168.2.1
device vlane ether //lane mtu 1500 192.168.55.1
mymachine> ip subnet
subnet vlane.home . 192.168.55.0 ff:ff:ff:00 vlane
subnet ether.home . 192.168.2.0 ff:ff:ff:00 ether
mymachine> ip subnet add vlane.home . 192.168.55.1 ff:ff:fc:0
mymachine> ip subnet
subnet vlane.home . 192.168.52.0 ff:ff:fc:00 vlane
subnet ether.home . 192.168.2.0 ff:ff:ff:00 ether
```

See also:
device
route

10.48 trace

Syntax:
```
trace [<option>]
```

Description:
Turns on an IP tracing option, or lists the available options. Note that tracing messages are written to background output, so with the standard console one must use the "event" commands to see them.
An option can be:
• One of various keywords. The details of just what tracing messages are enabled by each keyword are not documented here; one must examine the source if one really wants to know.
• An association number (see the "files" command). For a TCP association this turns on detailed tracing of events (including all packet transmission and reception) on that association; for a UDP association it has no effect. The "files" command shows (by appending "TRACE") whether each association has tracing enabled.
• An interface name (see the "device" command). This turns on tracing of every packet sent or received through the interface (one line per packet). The "device" command shows (by appending "TRACE") whether each interface has tracing enabled.
• "ip". This turns on tracing for all interfaces.
• "all". This turns on all tracing.
Note that "trace" does *not* display which associations and interfaces are being traced; one must use the "files" and "device" commands for that. The "trace" command is "hidden", not shown by "ip help". It is useful mainly for debugging and troubleshooting

Example:
```
mymachine> ip trace
ip: try trace - <assoc no> <i/f name> all ip errors resolve ipatm
atmarp
iploop arp ipether icmp udp tcp tcphdr tcpstate routes riptx riprx
names
ip: currently tracing nothing
mymachine> ip trace tcp
ip: currently tracing tcp
```

See also:
errors
untrace

10.49 untrace

Syntax:

```
untrace [<option>]
```

Description:
Turns off IP tracing options. The syntax is the same as for "trace"; in particular, "untrace all" turns off all tracing. The "trace" command is "hidden", not shown by "ip help".

See also:
noerrors
trace

10.50 uptime
Syntax:
```
uptime
```

Description:
Displays the time for which the ADIOS system has been running.

Example:
```
mymachine> ip uptime
up 8 hours 33 minutes
```

10.51 version
Syntax:
```
version
```

Description:
Displays the system version, ATM address, and MAC address. (An obsolescent option "ip" still exists, but "version ip" now displays misleading information and should not be used.)

Example:
```
mymachine> ip version
Modem version 5.00.0.4 (August 27 1998) (DEBUG)
ATM address: 47.00.83.10.a2.b2.c2.00.00.00.00.00.00.00.20.2b.00.00.38.00
MAC address: 0:20:2b:0:0:38
```

10.52 ?
Syntax:
```
?
? <cmd>
? all
```

Description:
The "?" command is simply a synonym for the "help" command, and behaves in the same way.

See also:
```
Help
```

11 NAT Console Commands

11.1 *ip nat*

Syntax:
ip nat add|delete <i/f name>

Description:
This command adds or removes NAT functionality from the named interface. The interface name is the name as listed by the **ip device** command. NAT should always be enabled only on the interface connecting to the public network, not the interface connecting to the private network.

Example:
copperjet> ip nat add ethernet

See also:

11.2 *nat event*

Syntax:
nat event [n]

Description:
This command displays or sets the current level of event tracing in the NAT process. Larger values of **n** result in more verbose trace output, for example:
Event level Output
1 Only show fatal errors, e.g. lack of system resources
2 Only show important information and problems
3 Show the creation of new sessions
4 Show trace output for discarded packets
5 Show trace output for all packets

All trace messages are printed as background output, and therefore will not be displayed asynchronously on the console unless the **event show** command has been issued.

Example:
copperjet> nat event
Event level: 1
copperjet> nat event 2

See also:
event show

11.3 *nat help*

Syntax:
nat help [command]

Description:
Lists the commands provided by the NAT console interface. If an optional command name is supplied, help on that command's usage is displayed.

See also:

This document is not officialy released and has to be treated as confidential.

11.4 nat interfaces

Syntax:
nat interfaces

Description:
The **nat interfaces** command displays the IP router ports on which NAT is currently enabled. For each of these, a status and IP address is listed. The IP address is discovered automatically from the IP stack. The status shows the user whether NAT is currently operational on that interface ("enabled"), or whether NAT is still waiting to find out the interface's IP address ("not ready").

Example:
```
copperjet> nat interfaces
Name Status IP address
ethernet enabled 194.129.40.2
ppp not ready
```

See also:

11.5 nat inbound

Syntax:
nat inbound list
nat inbound add <i/f> <port>/<proto> <new IP> [quiet]
nat inbound delete <#>
nat inbound flush

Description:
This command enables the user to list or to set up a series of rules, to determine what happens to coming traffic. By default all incoming packets, other that packets arriving in response to outgoing traffic, will be rejected.
The **nat inbound add** command allows packets arriving on a specific port and IP protocol to be forwarded to a machine on the private network. **<i/f>** is an interface name as shown by the **nat interface list** command; **<port>** is the destination UDP or TCP port number to match in the incoming traffic; **<proto>** is the IP protocol, either "udp" or "tcp"; **<new IP>** is the new IP address on the private network which the packet's destination IP address should be translated to. If a rule is added for an interface on which NAT is not enabled, the rule is added anyway but a warning is printed to alert the user to this fact. **quiet** is a special option which should not normally be issued at the console, and causes this warning to be suppressed. The **quiet** option is automatically added by NAT to when writing its configuration to flash; this is because when a system boots, the NAT process reads in these rules before IP has registered any interfaces. **Nat inbound list** shows the current rules for inbound traffic, including all the arguments passed to the **nat inbound add** command. **Nat inbound delete** removes a rule, where <#> is the rule number as shown by the **nat inbound list** command. **nat inbound flush** removes all the rules.

Example:
```
copperjet> nat inbound add ethernet 80/TCP 192.168.219.38
copperjet> nat inbound list
# Interface Port/Proto New IP address
1 ethernet 80/tcp 192.168.219.38
2 r1483 21/tcp 192.168.219.40
copperjet> nat inbound delete 2
```

See also:

11.6 nat info

Syntax:
nat info

Description:
This command displays the values of various parameters, which are defined in the module file, for example the session table size and the session timeouts. NAT's current memory usage is also displayed.

Example:
```
copperjet> nat info
Interface table size 1 (116 bytes)
Session table size per interface: 128 (6656 bytes)
Total: 6656 bytes
Hash table size per interface: 128 (512 bytes)
Total: 512 bytes
Fragment table size per interface: 32 (640 bytes)
Total: 640 bytes
Max queued buffers: 16
Fragment timeout: 30
Support for incoming fragments: enabled
Support for outgoing fragments: enabled
Session timeouts:
ICMP query: 10
UDP: 30
TCP (established): 300
TCP (other): 15
Initial port number: 10000
```

See also:
nat version

11.7 nat protocol

Syntax:
nat protocols

Description:
The **nat protocols** command lists the application level gateways (ALGs) provided in the current image in order to support particular higher-level protocols, and the port or ports which each ALG monitors.

Example:
```
copperjet> nat protocols
Name Port/IP protocol
ftp 21/tcp
```

See also:

11.8 nat sessions

Syntax:
nat sessions <i/f> [all | summary]

Description:
The **nat sessions** command displays a list of currently active NAT sessions on the interface

15-08-00   page 64 of 98

<i/f>. In this context, a session is a pair of source IP addresses and port numbers (and corresponding new port number) that NAT regards as one side of an active connection. For each TCP or UDP session active, the source and destination IP address and port number, and the local port number and the age of the session, are printed. The **all** option causes the **sessions** command to print out information on every session, including sessions which have timed out. Normally the **sessions** command only shows active sessions (those which have not timed out).

The **summary** command does not show detailed information on each session, but only prints out the total number of active, timed out and available sessions.

Example:
```
copperjet> nat sessions ppp
Proto Age NAT port Private address/port Public address/port
TCP 34 1024 192.168.219.38/3562 194.129.50.6/21
TCP 10 1025 192.168.219.64/2135 185.45.30.30/80
Total:
2 sessions active
101 sessions timed out
126 sessions available
```

See also:

11.9 nat stats

Syntax:
nat stats <i/f> [reset]

Description:
This command displays various statistics gathered by NAT on the interface **<i/f>**. These are cumulative totals since power on, or since the **reset** keyword was given. The **nat stats c**ommand does not provide the total number of packets or bytes transferred, as this information is normally available from the device driver on the interface which NAT is filtering.

Example:
```
copperjet> nat stats ethernet
Outgoing TCP sessions created: 456
Outgoing UDP sessions created: 123
Outgoing ICMP query sessions: 12
Outgoing ICMP errors: 0
Incoming ICMP errors: 6
Incoming connections refused: 2
Sessions deleted early: 0
Fragments currently queued: 0.
```

See also:

11.10 nat version

Syntax:
nat version

Description:
This command displays NAT's internal version number.

Example:
```
copperjet> nat version
NAT Version 1.00
```

See also:
nat info

11.11 nat dump

Syntax:
nat dump on|off

Description:
This command is only available in debug builds. **nat dump** causes a detailed dump of the information in each packet's header to be printed both before and after translation. This command is provided for debug purposes.

Example:
```
copperjet> nat dump on
```

See also:

11.12 nat fragments

Syntax:
nat fragments <i/f name>

Description:
This command is only available in debug builds. **nat fragments** prints information on the queues in which NAT holds fragmented IP datagrams, displaying the IP datagram identifier, the number of fragments queued and a NAT session pointer for each queue. This command is provided for debug purposes only.

Example:
```
copperjet> nat fragments ether
```

See also:

11.13 nat hashtable

Syntax:
nat hashtable <i/f name>

Description:
This command is only available in debug builds. **nat hashtable** prints the number of sessions linked to each entry in the hashtable used to look up outgoing packet on the given interface. This command is provided for debug purposes only.

Example:
```
copperjet> nat hashtable ethernet
# Linked sessions
0 1
1 0
2 1
3 2
```

See also:

This document is not officialy released and has to be treated as confidential.

12 Protocols requiring an Application Level Gateway

NAT is transparent to the majority of common protocols, including, but not limited to:
- DNS queries
- HTTP
- NNTP
- POP3
- SMTP
- SSH (most features)
- Telnet
- TFTP
- Windows drive sharing (most features)

However, there are certain other protocols, which cannot operate through a NAT-enabled router without special provision being made for them. These protocols include:
- CUSeeMe.
- DNS (some functions)
- Doom
- FTP
- H323.1
- ICQ
- IPSec
- IRC
- Net2Phone
- NetShow
- NTT AudioLink / NTT SoftwareVision
- Quake / Quake II
- RealAudio / RealVideo
- RSTP Applications
- RSVP
- StreamWorks
- Talk / ntalk True Speech
- VDOLive
- VIVOActive
- Vosaic
- VXtreme
- Yamaha MIDPlug

In order for applications using one of these protocols to operate transparently through a NAT-enabled router, in the majority of cases all that is required is the addition of an Application Level Gateway (ALG) for that protocol. Each ALG modifies the payloads of IP packets for that protocol. There is a minor issue when using Windows drive sharing to allow access to a computer on the private network from the public network. As well as configuring a rule to allow incoming traffic on the NetBIOS port, it is necessary to ensure that the computer name of the machine exporting the shared drive is the same as the name of the router interface on which NAT is enabled. This appears to be a limitation of the NetBIOS-over-TCP protocol. There are no problems accessing shared drives on the public network from a machine on the private network. Another class of protocols uses the TCP/UDP source port number to provide authentication. These protocols assume that connections originating from a source port number less than 1024 can be "trusted" without further verification. These include:
- LPR
- NFS
- RSH / RLOGIN

Since NAT normally translates all source port numbers to a high value, these protocols may fail to work. However, this should not cause any problems since, due to their inherent lack of security, these protocols are only used within a LAN. In the unlikely event that it is necessary to use these protocols, the value of NAT_FIRST_PORT+NAT_MAX_SESSIONS may be changed to less than 1024, but be aware that this will cause **all** sessions to originate from "privileged" ports. Currently, only an ALG for the FTP File Transfer Protocol is included as standard with ADIOS NAT.

13 DHCP Server Console Commands
This section describes console commands provided by the `dhcpserver` process.

13.1 *dhcpserver config*
Syntax:
dhcpserver config [add <text>|confirm|delete|flush]

Description:
This command displays or edits the current configuration of the DHCP server. To display current configuration, provide no arguments to the command. Use of the "add" argument adds the line `<text>` to the configuration file. Use of the "confirm" argument reparses the configuration file, confirming the changes made if the parse is successful. Use of the "delete" argument deletes the last line from the configuration file. Use of the "flush" argument deletes the whole configuration. Following any change to the configuration file, it is necessary to "confirm" the changes, issue a "flashfs update" to commit the change to FLASH, and then restart the system before the changes can take effect.

Example:
```
copperjet> dhcpserver config

Current DHCP server configuration allow unknown-clients;
allow bootp;
subnet 192.168.219.0 netmask 255.255.255.0 {
range 192.168.219.10 192.168.219.30;
max-lease-time 5000;
}
copperjet> dhcpserver config flush
Configuration file flushed.
copperjet> dhcpserver config

Current DHCP server configuration
(Issue "dhcpserver config confirm" followed by "flashfs update"
to confirm new configuration)
copperjet>
```

13.2 *dhcpserver help*
Syntax:
dhcpserver help <command|all>

Description:
This command provides help on the various console commands provided by the ADIOS DHCP server. Specifying a command name gives detailed help on the command. Specifying "all" gives detailed help on all available commands.

Example:
```
copperjet> dhcpserver help
Help is available on the following commands:
config help pool status trace untrace
```

13.3 *dhcpserver pool*
Syntax:
dhcpserver pool [verbose]

Description:
This command gives a summary of DHCP server memory usage. The verbose option shows the entire memory allocation/free list.

Example:
```
copperjet> dhcpserver pool
```
DHCP Server Memory Pool Status
total pool size 79968
free 52448
allocated 27520
mean alloc chunk 59
max free chunk 30416

13.4 dhcpserver status

Syntax:
dhcpserver status

Description:
This command provides a summary of all leases known to the server on each interface in turn. It also shows remaining available IP addresses (i.e. those with no specified lease time, or client identifier).

Example:
```
copperjet> dhcpserver status
DHCP Server Lease Status
Interface "ethernet"
IP address | Client UID | Expiry
---------------+----------------------+-----------------
192.168.219.1     | 01:00:20:af:20:6f:59 | 11 hours
192.168.219.2     | 01:00:20:af:11:2a:ac | 8 hours
192.168.219.3     | Myclient | 140 seconds
192.168.219.4     | 00:20:af:20:00:2b | 2 days
192.168.219.5     | <unknown> | Never
192.168.219.6     | <unknown> | Never
192.168.219.7     | <unknown> | Never
192.168.219.8     | <unknown> | Expired
192.168.219.9     | <unknown> | Expired
192.168.219.10    | Foobarbozzle | Expired
```

13.5 dhcpserver trace

Syntax:
dhcpserver trace <trace option>

Description:
This command enables or disables tracing for the DHCP server. If no arguments are given the command lists the current tracing options enabled. The following trace options are available:

lease   Report changes in lease status (any device)
bootp   Report any BOOTP interoperation/emulation
error   Report all errors (fatal events)
warn   Report all warnings
note   Report all note-level (minor) events
all   All trace options

Tracing options are disabled by using the `untrace` command in the same way.
Saving configuration does not preserve the current tracing options that are enabled. By default, only tracing of **error** is enabled.

Example:
```
copperjet> dhcpserver trace
No tracing options currently enabled.
copperjet> dhcpserver trace error warn note
Currently tracing: error warn note
```

13.6 dhcpserver version

Syntax:
dhcpserver version

Description:
This command displays the current version number of the ADIOS DHCP software.

Example:
```
copperjet> dhcpserver version
ADIOS DHCP Version 1.02
copperjet>
```

14 DHCP Client Console Commands

This section describes console commands provided by the `dhcpclient` process.

14.1 dhcpclient config

Syntax:
```
dhcpclient config
```

Description:
This command displays the current configuration of the DHCP client, including selected DHCP options.

Example:
```
Coperjet> dhcpclient config
---
DHCP client configuration file: '//isfs/dhclient.conf'
timeout 60;
retry 60;
reboot 10;
backoff-cutoff 40;
interface "ethernet" {
send dhcp-lease-time 5000;
send dhcp-client-identifier "Galapagos";
}
```

14.2 dhcpclient help

Syntax:
```
dhcpclient help <command|all>
```

Description:
This command provides help on the various console commands provided by the ATMOS DHCP client. Specifying the command name gives detailed help, and specifying the argument "all" gives detailed help on all commands.

Example:
```
bd2000> dhcpclient help
Help is available on the following commands:
config help pool status trace untrace
```

14.3 dhcpclient pool

Syntax:
```
dhcpclient pool [verbose]
```

Description:
This command displays the state of the memory pool being used by the DHCP client. Should the client ever run out of memory, use of this command is helpful in determining the optimum memory pool size for the client. For example, supporting DHCP client functionality on several interfaces simultaneously will require proportionately more memory. The default pool size specified in the system file `dhcpclient` is 40000 bytes.

The verbose option lists all allocated and freed memory chunks.

Example:
```
bd2000> dhcpclient pool
DHCP Client Memory Pool Status
total pool size 39968
```

15-08-00

```
free 21392
allocated 18576
mean alloc chunk 67
max free chunk 13904
```

14.4 dhcpclient status

Syntax:
```
dhcpclient status [all]
```

Description:
This command provides DHCP status information for the active bound lease associated with each valid interface in turn, including IP address, time until lease renewal, subnet mask and DHCP server address. Including the "all" flag shows, for each valid interface, the active lease, leases which are being, or have been offered to the interface, and any leases which are still being held by the client which are not currently active (since a single interface can only have one active lease at a time).

Example:
```
Copperjet> dhcpclient status

DHCP Client Lease Status (active lease only)
Interface 'ethernet'
Status | Server ID | IP address | Subnet mask | Renewal
---------+-----------------+----------------+----------------+------------
*ACTIVE* | 192.168.219.151 | 192.168.219.1 | 255.255.255.0 | 31 seconds
```

14.5 dhcpclient trace

Syntax:
```
dhcpclient trace <trace option>
```

Description:
This command enables or disables tracing for the DHCP client. If no arguments are given the command lists the current tracing options enabled.
The following trace options are available:

lease Report changes in lease status (any interface)
bootp Report any bootp interoperation
error Report all errors (fatal events)
warn Report "warn" level events (important events)
note Report "note" level events (minor/frequent events)
all All trace options

Tracing options are disabled by using the "untrace" command with the option names to be disabled. Saving configuration does not preserve the current tracing options that are enabled. By default tracing of **erro**r, **warn** and **note** are enabled.

Example:
```
Copperjet> dhcpclient trace
No tracing options currently enabled.
Copperjet> dhcpclient trace error warn note
Currently tracing: error warn note
```

This document is not officialy released and has to be treated as confidential.

15 DHCP-related IP process commands

The following commands are not provided by the DHCP client process but by the IP process
`ip.`

15.1 ip device

Syntax:
```
ip device add <i/f> <type> <file> [mtu <size>] [<IP address>|dhcp]
ip device
```

Description:
The **ip device add** command adds an interface to the configuration of the IP stack. The last
parameter of the command would normally the IP address of the interface; use of the string
dhcp causes the IP address to be discovered by the DHCP client software. Note that using
the flag **dhcp** on an interface precludes running a DHCP server on that interface!
The **ip device** command lists the current configuration of any devices attached to the IP
stack. A device configured to use DHCP will show "dhcp" in the "IP address" column,
followed by the actual IP address discovered and bound by DHCP, if any.
For interfaces configured to use DHCP, saving configuration only marks the interface as using
DHCP; it does not save the actual IP address discovered by DHCP, which must be renewed.
A useful method of automatically configuring suitable IP devices is to put a "device add"
statement into the file "//isfs/resolve" and downloading it upon booting the image.

Example:
```
Copperjet> ip device add ethernet ether //edd dhcp
…DHCP then discovers the IP address for the interface…
Copperjet> ip device
# type dev file IP address
device ethernet ether //edd mtu 1500
```

16 DHCP Relay Console Commands

This section describes console commands provided by the `dhcprelay` process.

16.1 dhcprelay add

Syntax:
dhcprelay add [ip address]

Description:
This command adds the entered IP address to the relay's list of known DHCP servers. Changes made will not come into effect until system restart. Ensure that you save configuration (using **flashfs updat**e) prior to restarting. A maximum of 10 DHCP server addresses can be stored by the relay.

Example:
`copperjet>` dhcprelay add 192.168.219.7
dhcprelay: Change will have no effect until 'flashfs update' and reboot.
copperjet>

16.2 dhcprelay config

Syntax:
dhcprelay config

Description:
This command displays the current configuration of the DHCP relay, which comprises a list of IP addresses of known DHCP servers.

Example:
```
copperjet> dhcprelay config
DHCP Relay - Registered DHCP Servers
192.168.219.6
copperjet>
```

16.3 dhcprelay delete

Syntax:
dhcprelay delete [ip address]

Description:
This command deletes the specified IP address from the relay's list of known DHCP servers, if the named server exists. If the address is omitted, then the last server address entry in the relay's list is deleted. Changes made will not come into effect until system restart. Ensure that you save configuration (using **flashfs updat**e) prior to restarting.

Example:
`copperjet>` dhcprelay delete 192.168.219.7
dhcprelay: Change will have no effect until 'flashfs update' and reboot.
copperjet>

16.4 dhcprelay help

Syntax:
dhcprelay help <command|all>

Description:

This command provides help on the various console commands provided by the ADIOS DHCP relay. Specifying the command name gives detailed help, and specifying the argument "all" gives detailed help on all commands.

Example:
```
copperjet> dhcprelay help
```

Help is available on the following commands:
config help pool status trace untrace

16.5 dhcprelay pool

Syntax:
dhcprelay pool [verbose]

Description:
This command displays the state of the memory pool being used by the DHCP relay. The verbose option lists all allocated and freed memory chunks.

Example:
```
copperjet> dhcprelay pool
DHCP Relay Memory Pool Status
total pool size 10000
free 9838
allocated 162
mean alloc chunk 162
max free chunk 9838
```

16.6 dhcprelay status

Syntax:
dhcprelay status

Description:
This command lists the interfaces upon which the DHCP relay entity is currently listening (if the relay has at least one valid DHCP server address in its list).

Example:
```
copperjet> dhcprelay status
DHCP Relay listening on:
Ethernet
```

16.7 dhcprelay trace/untrace

Syntax:
dhcprelay <trace|untrace> [trace options]

Description:
This command enables or disables tracing for the DHCP relay. If no arguments are given the command lists the current tracing options enabled.
The following trace options are available:
lease   Report changes in lease status (any interface)
bootp   Report any bootp interoperation
error   Report all errors (fatal events)
warn   Report "warn" level events (important events)
note   Report "note" level events (minor/frequent events)
all   All trace options

Tracing options are disabled by using the "untrace" command with the option names to be disabled. Saving configuration does not preserve the current tracing options that are enabled. By default tracing of **error** is enabled.

Example:
```
copperjet> dhcprelay trace
No tracing options currently enabled.

copperjet> dhcprelay trace error warn note
Currently tracing: error warn note
```

16.8 dhcprelay version
Syntax:
dhcprelay version

Description:
The `version` command displays the current ADIOS DHCP software version.

Example:
```
copperjet> dhcprelay version
ADIOS DHCP Version 1.07
copperjet>
```

17 DNS Relay Console Commands

This section describes console commands provided by the `dnsrelay` process.

17.1 dnsrelay config

Syntax:
dnsrelay config [reset]

Description:
This command displays the configuration of the DNS relay, including the DNS server address, the number of communication retries the relay will attempt in the event of a failed connection, and whether or not the relay has managed to connect successfully to a DNS server. Adding the keyword "reset" to the "config" command results in the configuration being reset to factory default settings.

Example:
```
copperjet> dnsrelay config
Server discovery mode : MANUAL
DNS Server address : 192.168.96.200 - Connected
Max connection retries: 3
copperjet> dnsrelay config reset
dnsrelay: Default settings restored. (Warning: Must re-connect to DNS
server,
dnsrelay: all old outstanding traffic and connections will be
dropped).
```

17.2 dnsrelay help

Syntax:
dnsrelay help [command|all]

Description:
This command provides help on the various console commands provided by the ADIOS DNS relay. Specifying the command name gives detailed help, and specifying the argument "all" gives detailed help on all commands.

Example:
```
copperjet> dnsrelay help
Valid DNS relay commands are:
config help pool retry server
status trace untrace version
```

17.3 dnsrelay pool

Syntax:
dnsrelay pool [verbose]

Description:
This command displays the state of the memory pool being used by the DNS relay.
The "verbose" option lists all allocated and freed memory chunks.

Example:
```
copperjet> dnsrelay pool
DNS Relay Memory Pool Status
total pool size 9968
free 9872
allocated 96
```

```
mean alloc chunk 32
max free chunk 9856
```

17.4 dnsrelay retry

Syntax:
dnsrelay retry <retry value>

Description:
This command sets the maximum number of retries the DNS relay is allowed to perform in the event of connection or transmission failure. The retry value must be a number between 1 and 10.

Example:
```
copperjet> dnsrelay retry 4
Connection retry value set to 4.
copperjet>
```

17.5 dnsrelay server

Syntax:
dnsrelay server <DNS server IP address>

Description:
This command tells the DNS relay which DNS server to contact. Caution must be exercised when using this command - if the DNS relay already knows which DNS server to contact then all existing connections will be reset, all outstanding traffic dropped, and the relay will then attempt to communicate with the newly appointed DNS server.

Example:
```
copperjet> dnsrelay server 192.168.219.50
DNS server address set to 192.168.219.50.
copperjet>
```

17.6 dnsrelay status

Syntax:
dnsrelay status

Description:
This command displays the status of the DNS relay, including whether or not it knows which DNS server to try to contact and, if so, whether or not it has successfully connected to the server.

Example:
```
copperjet> dnsrelay status
DNS relay status
DNS server address discovery incomplete.
copperjet> dnsrelay server 192.168.219.50
DNS server address set to 192.168.219.50.
copperjet> dnsrelay status
DNS relay status
DNS server address : 192.168.219.50
Connection status : Connected
copperjet>
```

17.7 dnsrelay trace/untrace

Syntax:
dnsrelay <trace|untrace> [trace options]

Description:
This command enables or disables tracing for the DNS relay. If no arguments are given the command lists the current tracing options enabled. The following trace options are available:
socket Report ALL socket-related I/O

query Trace DNS resolver queries
response Trace DNS server responses
error Report all serious, error-evel events
warn Report all minor, warning-level events
conn Trace DNS server connectivity
all Activate all trace options

Trace options are disabled by using the "untrace" command with the option names to be disabled. Saving configuration does not preserve the current tracing options that are enabled. By default tracing of **error** is enabled.

Example:
```
copperjet> dnsrelay trace
No tracing options currently enabled.

copperjet> dnsrelay trace error warn query
Currently tracing: error warn query
```

17.8 dnsrelay version

Syntax:
dnsrelay version

Description:
The version command displays the current ADIOS DNS relay software version.

Example:
```
copperjet> dnsrelay version
ADIOS DNS Relay Version 1.01
copperjet>
```

18 Changing from Ethernet to USB.

When the Copperjet is set to operate as a USB device, it searches both the USB and Ethernet interfaces for a server to boot from. (This means, when set as a USB device, the Copperjet can boot from Ethernet as well, but DHCP service on the Ethernet would be required.) In the case of USB usage, normally the Ethernet interface will not be used, and the Copperjet will find its boot server on the USB bus. This is the USB host. The USB drivers on the PC (host) automatically take care of booting the Copperjet. No special service is required.

To switch the Copperjet from Ethernet to USB, the Copperjet must be set to boot from the USB or Ethernet interface instead of flash. The following steps describe the way to do so:

1.   Install the USB drivers on your PC (USB host).

2.   Connect the Copperjet through a serial cable to a standard terminal (Hyperterminal) using these settings: 9600, 8, N, 1, flow control off.

3.   Connect the Copperjet through a USB cable to the USB host (This may be the same PC running the terminal, but this is not required).

4.   In the terminal, hold down the '*' key (shift-8) while powering on the Copperjet.

5.   As soon as the question "Boot from Ethernet, USB or Flash? (E/U/F)" comes up, press "U" for USB boot.

6.   As soon as the lines with dots appear, which indicate the progress of the boot process, hold down the space bar until you enter the special command console.

7.   Press *<enter>* to start a new line, and enter the command:

        configeeprom serialboot yes *<enter>*

8.   The Copperjet is now ready to boot over USB. Power off the Copperjet and power if on again.

19 Changing from USB to Ethernet.

When booting the Copperjet from Ethernet (with DHCP), the boot image determines what the functionality of the Copperjet becomes. This way an Ethernet bridge could be set up. However, it is not desirable for a stand-alone device like an Ethernet bridge to require a DHCP server to start up. This would put an unacceptable demand on end users. Therefore, when the Copperjet is set to operate as an Ethernet device, it should boot from flash. Obviously, a flash boot can only be successful if the flash has been programmed properly. If the flash is empty, only an Ethernet boot with DHCP can be used to program the flash. Refer to the flashfs documentation for further details.

To switch the Copperjet from USB to Ethernet, the Copperjet must be set to boot from flash instead of the USB or Ethernet interface. The following steps describe the way to do so:

1.   Connect the Copperjet through a serial cable to a standard terminal (Hyperterminal) using these settings: 9600, 8, N, 1, flow control off.

2.   Connect the Copperjet through a USB cable to the USB host (This may be the same PC running the terminal, but this is not required).

3.   Power on the Copperjet.

4.   As soon as the lines with dots appear, which indicate the progress of the boot process, hold down the space bar until you enter the special command console.

5.   Press *<enter>* to start a new line, then enter the command:

```
configeeprom serialboot no <enter>
```

6.   The Copperjet is now ready to boot from flash. Power off the Copperjet and power it on again.

20 Phy Console commands

Console commands should be prefixed with `phy` in order to direct them to the **phy** process.
Below is a list of currently available **phy** commands:

```
bat
binary
dspcommand
perf
status
setled
version
```

20.1 bat

Syntax:
`bat`

Help text:
Display the Bit Allocation Table.

Description:
Displays the utilized bit allocation for each single DMT band in the up- and downstream. The band number multiplied by 4.3125 Hz gives the start frequency of the each DMT band.

Example:

```
DS:    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19

  0:    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 20:    3  3  4  4  6  6  6  7  8  9  9  9 10  9  9 10 11 11 11 11
 40:   11 10 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
 60:   11 11 11 11  0 11 11 11 11 11 11 11 11 10 11 10 10 10 11 11
 80:   10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
100:   10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
120:   10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  9  9  9  9
140:    9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
160:    9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
180:    9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
200:    9  9  9 10  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
220:    9  9  9  9  9  9  9  9  9  9  9  9 10  9  9  9  9  9  9  9
240:    9  9  9  9  9  9  9  9  9  9  8  8  7  7  6  4
                                                 Total 2208

US:    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19

  0:    0  0  0  0  0  0  0  0  0  3  4  6  7  9 10 11  0 12 12 12
 20:   12 12 12 12 12 12 12 12 11 11 11  9
                                                 Total  224
```

20.2 perf

Syntax:
```
perf window <size>
perf reset
perf show
perf start
perf stop
```

Help text:
Performance monitoring commands.

Description:
This command is used to display, either once or repeatedly, basic ADSL performance counters. In contrast with the equally named parameters listed from the `status` command, these parameters are cumulative. These values are updated once every performance count window. The performance count window is the time, measured in super frames, in which the non-cumulative DSP performance counters are updated. (see the command `status`). After such an update, the non-cumulative `(status)` counters are added to the current values of the cumulative `(perf)` counters.

Below is a list of displayed performance counters:

CRCI_Count	Count of CRC-8 anomalies on the downstream interleave path.
CRCF_Count	Count of CRC-8 anomalies on the downstream fast path.
FECI_Count	Count of FEC-I anomalies on the downstream interleave path.
FECF_Count	Count of FEC-F anomalies on the downstream fast path.
ECS_Count	Count of 1-second intervals with one or more downstream FEC anomalies.
ES_Count	Count of 1-second intervals with one or more downstream CRC-8 anomalies, or one or more LOS defects, or one or more SEF defects, or one or more LPR defects.
SES_Count	Count of 1-second intervals with 18 or more downstream CRC-8 anomalies, or one or more LOS defects, or one or more SEF defects, or one or more LPR defects.
LOSS_Count	Count of 1-second intervals containing one or more downstream LOS defects.
UAS_Count	Count of 1-second intervals for which the ADSL line is unavailable.
HECI_Count	Count of HEC-I anomalies on the downstream interleave path.
HECF_Count	Count of HEC-F anomalies on the downstream fast path
TotHECI_Count	Count of the total number of downstream cells passed through the cell delineation process operating on the interleaved data while in the SYNC state.
TotHECF_Count	Count of the total number of downstream cells passed through the cell delineation process operating on the fast data while in the SYNC state.
UserCellI_Count	Count of the total number of downstream cells in the interleave path delivered at the T-R interface.
UserCellF_Count	Count of the total number of downstream cells in the fast path delivered at the T-R interface.
IdleErrI_Count	Count of the number of bit errors in the idle cell payload received on the interleave path.
IdleErrF_Count	Count of the number of bit errors in the idle cell payload received on the fast path.

Subcommands:
Show	The subcommand `show` is used to print the list only once.
Start	The subcommand `start` is used to start printing the list automatically after every performance count window. (default: off.)
Stop	The subcommand `stop` is used to stop automatic printing.
Reset	The subcommand `reset` is used to reset the cumulative values.
Window	The subcommand `window` is used to redefine the size of the performance count window. (default: 1764 superframes)

Example:
```
192.168.1.107 phy> perf show

  CRCI_Count:     11       SES_Count:      0     TotHECF_Count:     0
  CRCF_Count:      0      LOSS_Count:      0   UserCellI_Count:     0
  FECI_Count:     11       UAS_Count:      0   UserCellF_Count:     0
  FECF_Count:      0      HECI_Count:      0    IdleErrI_Count:     0
   ECS_Count:      0      HECF_Count:      0    IdleErrF_Count:     0
    ES_Count:      0   TotHECI_Count:      0
```

20.3 status

Syntax:
```
status
```

Help text:
Status info from the phy.

Description:
Prints a list of parameters from the DSP binary. A detailed description of the meaning of these parameters is out of the scope of this document.

20.4 setled

Syntax:
```
setled <mask>
```

Help text:
Set the state of the frontpanel LEDs. (specify hex mask)

Description:
Sets the state of the front panel LEDs according to the specified hexadecimal bit mask.
The "G.DMT" LED is driven by bit 1 and the Power LED is driven by bit 14. Bit 0 and bit 15 are reserved. A logical "1" in the mask switches the LEDs on, except for the power LED which is inverted.

20.5 version

Syntax:
```
version
```

Help text:
Version information from the DSP binary.

Description:
Displays the version information that was extracted from the DSP binary.

Example:
```
192.168.1.107 phy> version
version info of DSP binary: 01 10 May 20 2000 21:11:56    01
```

21 TFTP Console Commands

The TFTP process provides a number of console commands. Some commands are available whatever the build configuration, whilst some commands are available only if client mode operation is enabled. Console commands should be prefixed with `TFTP` in order to direct them to the **TFTP** process.

To aid understanding, an indication is made where a command is client mode only.

21.1 connect

Syntax:
```
connect <node_name> || <ipaddr>
```

Scope:
Client mode only.

Description:
The 'connect' command is used to specify the remote host name or IP address that will be used in subsequent client mode transfers.
Either a host name may be entered, searched for in the 'ipaddresses' configuration file, or an IP address in the form 'abc.def.ghi.jkl'. If the host name is not recognised or the IP address does not convert correctly an error is signalled.
The non-appearance of an error message after the command *does not* signify that the remote host is accessible, only that the syntax of the command was appropriate.
This command is required before a client mode user first attempts to 'put' or 'get' a file, but need not be issued again unless its desired to change the remote machine name or address.

Example:
```
connect 192.168.200.10
```

See also:
```
put, get
```

21.2 get

Syntax:
```
get <remote_file> [local_file]
```

Scope:
Client mode only.

Description:
The 'get' command requests TFTP to retrieve a file from the remote host previously specified using the 'connect' command.
Only files that fit within the file storage area within the session data (currently 8K) can be retrieved. This means that it not possible to initiate a software update from the client.
By default the file is named locally as the remote filename but by specifying a second filename an implicit rename is performed.

Example:
```
get ipaddresses
```

See also:
```
connect, put
```

21.3 help

Syntax:
help

Description:
The 'help' command displays the help text which lists the (commonly used) TFTP commands. The 'init' command is not listed in the help text.
The trace command has a large number of optional parameters and detail on this command may be displayed by typing 'trace help'.
If the software build supports client mode operation, these commands will be displayed in the help text.

Example:
help

See also:
version, trace help

21.4 init

Syntax:
init

Description:
The 'init' command causes all sessions to be initialised to an idle state. This command can be used during testing but is not required in normal operation. The command does not appear in the help text.

Example:
init

21.5 list

Syntax:
List

Description:
The 'list' command displays the status of any active sessions. This command is primarily intended for use during debug.

Example:
list

21.6 put

Syntax:
put [local_file] <remote_file>.VIRATA TFTP DO-007137-PS

Scope:
Client mode only.

Description:
The 'put' command requests TFTP to transmit a file to the remote host previously specified using the 'connect' command.

By default the file is named remotely as the local filename but by specifying a second filename an implicit rename is performed.

Example:
`put ipaddresses`

See also:
`connect, get`

21.7 trace

Syntax:
`trace <help> || <-*> || <+event1> <-event2>`

Description:
The 'trace' command allows the user to examine the currently set trace types or add /subtract trace types. Trace help lists all the available tracing types.
If the trace command is used with no parameters the currently set trace types are displayed.

Example:
`trace +tmr_exp`

21.8 version

Syntax:
`version`

Description:
The 'version' command displays software version information about the process.
The version number, which is displayed in the form 'a.bc', is defined in the module file as an integer 'abc'.

Example:
`version`

See also:
`Help`

22 PPTP Console Commands

Console commands should be prefixed with `pptp` in order to direct them to the **pptp** process.

22.1 Console object types

The **pptp** process provides a number of PPTP connection *tunnel*s. A tunnel consists of a control connection between the local PAC and a PNS, and a data connection (known as a call) through which a number of PPP connections or *channels* may be multiplexed.
The upper limits of several parameters may be configured using the '`config resource`' console command. The current state of each tunnel is saved by '`config save`'.

22.2 Console examples

These examples are for configuration of the PPTP Access Concentrator (PAC). Obviously the PPP client or server and the PNS must also be configured.

22.2.1 Dial-Out

The PPTP module uses functionality provided by the PPP module. Configure PPP channel 2 for an outgoing PPTP connection, using PPTP tunnel 1, and using PVC 800.

```
ppp 2 pvc 800
ppp 2 interface 0
ppp 2 tunnel 1 pptp out
ppp 2 enable
```

Next, configure the PPTP module to bind to an Ethernet interface with an IP address of, for example 192.168.10.1, and set up tunnel 1 to listen (waiting for the PNS to initiate the connection):

```
pptp bind 192.168.10.1
pptp 1 create listen
```

22.2.2 Dial-In

The PPTP module uses functionality provided by the PPP module. Configure PPP channel 2 for an incoming PPTP connection, using PPTP tunnel 1, and using PVC 800.

```
ppp 2 pvc 800 listen
ppp 2 interface 0
ppp 2 tunnel 1 pptp in
ppp 2 enable
```

Next, configure the PPTP module to bind to an Ethernet interface with an IP address, for example 192.168.10.1, and set up tunnel 1with the PAC initiating the connection: to a PNS with IP address, for example, 192.168.10.2.

```
pptp 1 bind 192.168.10.1
pptp 1 create 192.168.10.2
```

22.3 Console commands

The rest of this section details the individual console commands provided.

22.4 *bind*

Syntax:
`bind <ipaddress>|any|none`

Description:
Specify which local interface to bind a listener to for incoming control connections.
If `ipaddress` is specified, PPTP will listen on port 1723 *on that interface only* for incoming control connections. Typically this will be the IP address of the local side network interface.
If `any` is specified, PPTP will accept control connections on any interface.
If `none` is specified, no incoming control connections will be accepted; in this case, tunnels may only be established via the local `create` and `connect` commands.
Configuration saving saves this information. The default is `none`.

Example:
`pptp bind 192.168.1.1` listen for incoming control connections on
local interface 192.168.1.1 only

See also:
`<tunnel> create listen`

Notes:
An incoming connection can only be accepted if the listener has a free tunnel object allocated to it. (Such objects are allocated with the `<tunnel> create listen` command.) The tunnel object used will be freed for use again when the tunnel is closed by either end.

22.5 *<tunnel> connect*

Syntax:
`<tunnel> connect`

Description:
Explicitly connect a tunnel (that was created using `create <ipaddress>`) to the remote PNS that `create` specified, establishing the control connection.

Example:
`pptp 1 connect` connect tunnel 1 to configured PNS

See also:
`<tunnel> create <ipaddress>`
`<tunnel> disconnect` .ATMOS PPTP DO-007352-PS

Notes:
This command is meaningless if applied to a tunnel object that is allocated to the listener (as created with the `<tunnel> create listen` command); in this case it will produce an error message.

<tunnel> create

Syntax:
`<tunnel> create <ipaddress>|listen`

Description:
Create a tunnel object.
If `ipaddress` is specified, the tunnel is associated with a remote PNS at that IP address. The control connection is not actually established until use of the tunnel is requested by PPP, or an explicit `connect` is issued.

If `listen` is specified, the tunnel is allocated for use by an incoming control connection from a remote PNS. At least one such tunnel must exist if any incoming connections are to be accepted at all.

Incoming connections are mapped to the first available listening tunnel object. It is not currently possible to use properties of the incoming connection (such as its IP address, or information supplied in the fields of the PPTP control messages) to map the connection to a specific tunnel. Configuration saving saves this information. By default no tunnels are created.

Example:
`pptp 1 create 192.168.1.2` tunnel 1 connects to PNS at 192.168.1.2

See also:
```
<tunnel> connect
<tunnel> delete
```

22.6 *<tunnel> delete*

Syntax:
`<tunnel> delete`

Description:
Delete a tunnel object (the opposite of `create`). If the tunnel is currently connected, any active data connections across the tunnel are terminated and the control connection is closed.

Example:
`pptp 1 delete` delete tunnel 1

See also:
```
<tunnel> create <ipaddress>
```

22.7 *<tunnel> disconnect*

Syntax:
`<tunnel> disconnect`. ATMOS PPTP DO-007352-PS

Description:
Explicitly disconnect a tunnel (the opposite of `connect`). All data connections across the tunnel are terminated and the control connection is closed. If the tunnel object is associated with a particular remote PNS (as created with `<tunnel> create <ipaddress>`), it may be reconnected later, either explicitly with another `connect` command, or implicitly by PPP requesting to use it. If the tunnel object is allocated to the listener (as created with `<tunnel> create listen`), it is freed for use by future incoming connections.

Example:
`pptp 1 disconnect` disconnect tunnel 1

See also:
```
<tunnel> connect
```

22.8 *<tunnel> event*

Syntax:
`<tunnel> event [<n>]`

Description:

Read or set the trace output level for a tunnel.
Configuration saving does not save this value. The default event level is 1.
Event levels are:
1 only very serious errors reported (default)
2 definite protocol errors or very significant events reported
3 channels going up/down are reported
4 every packet and significant state change is reported
5 every packet sent/received is disassembled, and hex dumped

22.9 *<tunnel> info*

Syntax:
```
<tunnel> info [all]
```

Description:
Provide information about the current settings of this tunnel. This includes all configured state, and also current protocol information. Specifying 'all' prints out more information.
`info` and `status` are synonyms.

22.10 *list*

Syntax:
```
List
```

Description:
Lists all currently created tunnel objects and the IP address of the remote PNS associated with each one.

22.11 *version*

Syntax:
```
Version
```

Description:
Provide the version number for the source of the **pptp** process.

23 General TCP/IP information

The network numbers listed here are used as Internet addresses by the Internet Protocol (IP) [33,62]. The IP uses a 32-bit address field and divides that address into a network part and a "rest" or local address part. The division takes 3 forms or classes.

The first type of address, or class A, has a 7-bit network number and a 24-bit local address. The highest-order bit is set to 0. This allows 128 class A networks.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   NETWORK   |                Local Address                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
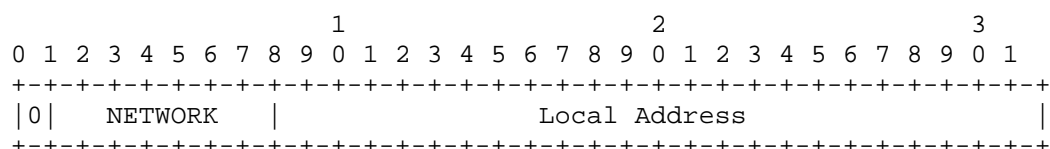
              Class A Address
```

The second type of address, class B, has a 14-bit network number and a 16-bit local address. The two highest-order bits are set to 1-0. This allows 16,384 class B networks.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0|          NETWORK           |          Local Address      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
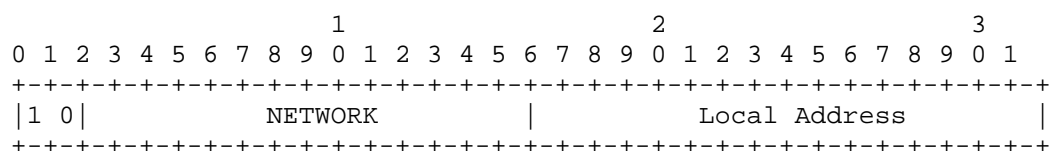
              Class B Address
```

The third type of address, class C, has a 21-bit network number and a 8-bit local address. The three highest-order bits are set to 1-0-0. This allows 2,097,152 class C networks.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 0|                  NETWORK                | Local Address |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
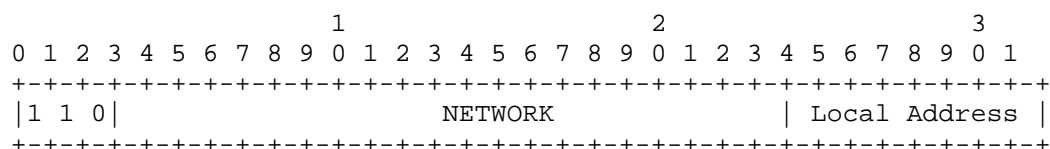
              Class C Address
```

One commonly used notation for internet host addresses divides the 32-bit address into four 8-bit fields and specifies the value of each field as a decimal number with the fields separated by periods. This is called the "dotted decimal" notation. For example, the internet address of ISIF in dotted decimal is 010.002.000.052, or 10.2.0.52.

The dotted decimal notation will be used in the listing of assigned network numbers. The class A networks will have nnn.rrr.rrr.rrr, the class B networks will have nnn.nnn.rrr.rrr, and the class C networks will have nnn.nnn.nnn.rrr, where nnn represents part or all of a network number and rrr represents part or all of a local address or rest field.

For various reasons, the assigned numbers of networks are sometimes changed. To ease the transition the old number will be listed as well. These "old number" entries will be marked with a "T" following the number and preceeding the name.

To reflect the allocation of network identifiers among various categories (see Appendix A), a one-character code is placed to the left of the network number (in the column marked by an asterisk): R for Research and Development, D for DoD, and C for Commercial.

23.1 Assigned Network Numbers

```
Class A Networks

* Internet Address   Name        Network            References
  ----------------   ----        -------            ----------
  000.rrr.rrr.rrr                 Reserved            [JBP]
R 001.rrr.rrr.rrr T BBN-PR        BBN Packet Radio Network  [JAW3]
R 002.rrr.rrr.rrr T SF-PR-1       SF-1 Packet Radio Network  [JEM]
R 003.rrr.rrr.rrr   RCC-NET       BBN RCC Network     [JGH]
R 004.rrr.rrr.rrr   SATNET        Atlantic Satellite Network[DM11]
D 005.rrr.rrr.rrr T DEMO-PR-1     Demo-1 Packet Radio Network[LCS]
R 006.rrr.rrr.rrr T SF-PR-2       SF-2 Packet Radio Network  [JEM]
  007.rrr.rrr.rrr                 Unassigned          [JBP]
R 008.rrr.rrr.rrr   BBN-NET       BBN Network         [JGH]
D 009.rrr.rrr.rrr T BRAGG-PR      Ft. Bragg Packet Radio Net [JEM]
R 010.rrr.rrr.rrr   ARPANET       ARPANET             [17,1,REK2]
R 011.rrr.rrr.rrr T UCLNET        University College London   [PK]
  012.rrr.rrr.rrr                 Unassigned          [JBP]
  013.rrr.rrr.rrr                 Unassigned          [JBP]
C 014.rrr.rrr.rrr   PDN           Public Data Network  [REK2]
  015.rrr.rrr.rrr                 Unassigned          [JBP]
  016.rrr.rrr.rrr                 Unassigned          [JBP]
  017.rrr.rrr.rrr                 Unassigned          [JBP]
R 018.rrr.rrr.rrr   MIT           MIT Network         [10,43,NC3]
  019.rrr.rrr.rrr                 Unassigned          [JBP]
  020.rrr.rrr.rrr                 Unassigned          [JBP]
D 021.rrr.rrr.rrr   EDN           DCEC EDN            [EC5]
  022.rrr.rrr.rrr                 Unassigned          [JBP]
R 023.rrr.rrr.rrr   MITRE         MITRE Cablenet      [44,APS]
  024.rrr.rrr.rrr                 Unassigned          [JBP]
R 025.rrr.rrr.rrr   RSRE-PPSN     RSRE / PPSN          [NM]
D 026.rrr.rrr.rrr   MILNET        MILNET              [HH6]
R 027.rrr.rrr.rrr   NOSC-LCCN     NOSC / LCCN         [KTP]
R 028.rrr.rrr.rrr   WIDEBAND      Wide Band Satellite Net  [CJW2]
  029.rrr.rrr.rrr                 Unassigned          [JBP]
R 030.rrr.rrr.rrr   DCN-UCL       UCL DCNET            [PK]
  031.rrr.rrr.rrr                 Unassigned          [JBP]
R 032.rrr.rrr.rrr   UCL-TAC       UCL TAC              [PK]
  033.rrr.rrr.rrr                 Unassigned          [JBP]
  034.rrr.rrr.rrr                 Unassigned          [JBP]
R 035.rrr.rrr.rrr   RSRE-NULL     RSRE Null Network    [NM]
R 036.rrr.rrr.rrr T SU-NET        Stanford University Network[JCM]
  037.rrr.rrr.rrr                 Unassigned          [JBP]
  038.rrr.rrr.rrr                 Unassigned          [JBP]
R 039.rrr.rrr.rrr   SRINET        SRI Local Network    [GEOF]
  040.rrr.rrr.rrr                 Unassigned          [JBP]
R 041.rrr.rrr.rrr   BBN-LN-TEST   BBN Local Network Testbed  [KTP]
  042.rrr.rrr.rrr                 Unassigned          [JBP]
  043.rrr.rrr.rrr                 Unassigned          [JBP]
R 044.rrr.rrr.rrr   AMPRNET       Amateur Radio Experiment Net[HM]
R 045.rrr.rrr.rrr T C3-PR         Testbed Development PRNET  [BG5]
R 046.rrr.rrr.rrr   UCB-ETHER     UC Berkeley Ethernet  [SXL]
R 047.rrr.rrr.rrr T SAC-PR        SAC Packet Radio Network  [BG5]
R 048.rrr.rrr.rrr   NDRE-TIU      NDRE-TIU            [PS3]
  049.rrr.rrr.rrr                 Unassigned          [JBP]
R 050.rrr.rrr.rrr   NDRE-RING     NDRE-RING           [PS3]
```

```
   051.rrr.rrr.rrr                  Unassigned              [JBP]
R 052.rrr.rrr.rrr T ROCKWELL-PR     Rockwell Packet Radio Net [EHP]
   053.rrr.rrr.rrr-126.rrr.rrr.rrr Unassigned              [JBP]
   127.rrr.rrr.rrr                  Reserved                [JBP]
```

Class B Networks

```
* Internet Address   Name         Network              References
  ----------------   ----         -------              ----------
   128.000.rrr.rrr                 Reserved                [JBP]
R 128.001.rrr.rrr    BBN-TEST-B    BBN-GATE-TEST-B         [RH6]
R 128.002.rrr.rrr    CMU-NET       CMU-Ethernet            [HDW2]
R 128.003.rrr.rrr    LBL-CSAM      LBL-CSAM-RESEARCH       [MO1]
R 128.004.rrr.rrr    DCNET         LINKABIT DCNET          [DLM1]
R 128.005.rrr.rrr    FORDNET       FORD DCNET              [DLM1]
R 128.006.rrr.rrr    RUTGERS       RUTGERS                 [CLH3]
R 128.007.rrr.rrr    DFVLR         DFVLR DCNET Network     [HDC1]
R 128.008.rrr.rrr    UMDNET        Univ of Maryland DCNET  [DLM1]
R 128.009.rrr.rrr    ISI-NET       ISI Local Network       [CMR]
R 128.010.rrr.rrr    PURDUE-CS     Purdue Computer Science [CXK]
R 128.011.rrr.rrr    BBN-CRONUS    BBN DOS Project      [12,WIM]
R 128.012.rrr.rrr    SU-NET        Stanford University Net [JCM]
D 128.013.rrr.rrr    MATNET        Mobile Access Terminal Net[DM11]
R 128.014.rrr.rrr    BBN-SAT-TEST  BBN SATNET Test Net     [DM11]
R 128.015.rrr.rrr    S1NET         LLL-S1-NET              [EAK1]
R 128.016.rrr.rrr    UCLNET        University College London [PK]
   128.017.rrr.rrr                 Unassigned              [JBP]
   128.018.rrr.rrr                 Unassigned              [JBP]
   128.019.rrr.rrr                 Unassigned              [JBP]
   128.020.rrr.rrr                 Unassigned              [JBP]
R 128.021.rrr.rrr    SF-PR-1       SF-1 Packet Radio Network [JEM]
R 128.022.rrr.rrr    SF-PR-2       SF-2 Packet Radio Network [JEM]
R 128.023.rrr.rrr    BBN-PR        BBN Packet Radio Network [JAW3]
R 128.024.rrr.rrr    ROCKWELL-PR   Rockwell Packet Radio Net [EHP]
D 128.025.rrr.rrr    BRAGG-PR      Ft. Bragg Packet Radio Net [JEM]
D 128.026.rrr.rrr    SAC-PR        SAC Packet Radio Network [BG5]
D 128.027.rrr.rrr    DEMO-PR-1     Demo-1 Packet Radio Network[LCS]
D 128.028.rrr.rrr    C3-PR         Testbed Development PR NET [BG5]
   128.029.rrr.rrr-191.254.rrr.rrr Unassigned              [JBP]
   191.255.rrr.rrr                 Reserved                [JBP]
```

Class C Networks

```
* Internet Address   Name         Network              References
  ----------------   ----         -------              ----------
   192.000.000.rrr                 Reserved                [JBP]
R 192.000.001.rrr    BBN-TEST-C    BBN-GATE-TEST-C         [RH6]
   192.000.002.rrr-192.000.255.rrr Unassigned              [JBP]
R 192.001.xxx.rrr-192.004.xxx.rrr BBN local networks       [SGC]
R 192.005.001.rrr    CISLNET       CISL Multics Network    [CH2]
R 192.005.002.rrr    WISC          Univ of Wisconsin Madison [RS23]
C 192.005.003.rrr    HP-DESIGN-AIDS HP Design Aids         [NXK]
C 192.005.004.rrr    HP-TCG-UNIX   Hewlett Packard TCG Unix [NXK]
D 192.005.005.rrr    BRLNET        BRLNET               [1,MJM2]
D 192.005.006.rrr    MINET         MINET                 [1,DHH]
R 192.005.007.rrr    CIT-CS-NET    Caltech-CS-Net        [65,DSW]
R 192.005.008.rrr    WASHINGTON    University of Washington [JAR4]
R 192.005.009.rrr    AERONET       Aerospace Labnet       [9,LCN]
R 192.005.010.rrr    ECLNET        USC-ECL-CAMPUS-NET      [MXB]
```

```
R 192.005.011.rrr    CSS-RING         SEISMIC-RESEARCH-NET         [RR2]
R 192.005.012.rrr    UTAH-NET         UTAH-COMPUTER-SCIENCE-NET    [RF1]
  192.005.013.rrr                     Unassigned                   [JBP]
  192.005.014.rrr                     Unassigned                   [JBP]
  192.005.015.rrr                     Unassigned                   [JBP]
  192.005.016.rrr                     Unassigned                   [JBP]
  192.005.017.rrr                     Unassigned                   [JBP]
  192.005.018.rrr                     Unassigned                   [JBP]
  192.005.019.rrr                     Unassigned                   [JBP]
  192.005.020.rrr                     Unassigned                   [JBP]
D 192.005.021.rrr    BRLNET1          BRLNET1                      [1,MJM2]
D 192.005.022.rrr    BRLNET2          BRLNET2                      [1,MJM2]
D 192.005.022.rrr    BRLNET3          BRLNET3                      [1,MJM2]
D 192.005.022.rrr    BRLNET4          BRLNET4                      [1,MJM2]
D 192.005.022.rrr    BRLNET5          BRLNET54                     [1,MJM2]
  192.005.026.rrr-223.255.254.rrr Unassigned                      [JBP]
  223.255.255.rrr                     Reserved                     [JBP]

Other Reserved Internet Addresses

Internet Address  Name            Network               References
----------------  ----            -------               ----------
224.000.000.000-255.255.255.255 Reserved                [JBP]

Network Totals

Assigned

Class          A         B         C      Total
Research      26        19      1033       1078
Defense        4         5         7         16
Commercial     1         0         2          3
Total         31        24      1042       1097

Maximum Allowed

Class          A         B         C      Total
Research       8      1024     65536      66568
Defense       24      3072    458752     461848
Commercial    94     12286   1572862    1585242
Total        126     16382   2097150    2113658
```

23.2 ASSIGNED PORT NUMBERS

Ports are used in the TCP [34,62] to name the ends of logical connections, which carry long term conversations. For the purpose of providing services to unknown callers a service contact port is defined. This list specifies the port used by the server process as its contact port. The contact port is sometimes called the "well-known port".

To the extent possible these same port assignments are used with UDP [42,62].

The assigned ports use a small portion of the possible port numbers. The assigned ports have all except the low order eight bits cleared to zero. The low order eight bits are specified here.

```
Port Assignments:
```

```
Decimal   Octal    Description                          References
-------   -----    -----------                          ----------
1         1        Old Telnet                           [40,JBP]
3         3        Old File Transfer                    [27,11,24,JBP]
5         5        Remote Job Entry                     [6,17,JBP]
7         7        Echo                                 [35,JBP]
9         11       Discard                              [32,JBP]
11        13       Who is on or SYSTAT                  [JBP]
13        15       Date and Time                        [JBP]
15        17       Who is up or NETSTAT                 [JBP]
17        21       Short Text Message                   [JBP]
19        23       Character generator or TTYTST        [31,JBP]
20        24       File Transfer (Default Data)         [36,62,JBP]
21        25       File Transfer (Control)              [36,62,JBP]
23        27       Telnet                               [41,62,JBP]
25        31       SMTP                                 [54,62,JBP]
27        33       NSW User System FE                   [14,RHT]
29        35       MSG ICP                              [29,RHT]
31        37       MSG Authentication                   [29,RHT]
33        41       Unassigned                           [JBP]
35        43       IO Station Spooler                   [JBP]
37        45       Time Server                          [22,JBP]
39        47       Unassigned                           [JBP]
41        51       Graphics                             [46,17,JBP]
42        52       Name Server                          [38,62,JBP]
43        53       WhoIs                                [57,62,JAKE]
45        55       Message Processing Module (receive)  [37,JBP]
46        56       MPM (default send)                   [37,JBP]
47        57       NI FTP                               [50,SK]
49-53     61-65    Unassigned                           [JBP]
55        67       ISI Graphics Language                [3,RB6]
57        71       Unassigned                           [JBP]
59        73       Augment File Mover                   [WWB]
61        75       NIMAIL                               [56,SK]
63        77       Unassigned                           [JBP]
65        101      Unassigned                           [JBP]
67        103      Datacomputer at CCA                  [8,JZS]
69        105      Trivial File Transfer                [47,62,KRS]
71        107      NETRJS                               [5,17,RTB]
72        110      NETRJS                               [5,17,RTB]
73        111      NETRJS                               [5,17,RTB]
74        112      NETRJS                               [5,17,RTB]
75        113      Unassigned                           [JBP]
77        115      any private RJE server               [JBP]
79        117      Name or Finger                       [23,17,KLH]
81        121      HOSTS2 Name Server                   [EAK1]
83        123      MIT ML Device                        [DPR]
85        125      MIT ML Device                        [DPR]
87        127      any terminal link                    [JBP]
89        131      SU/MIT Telnet Gateway                [MRC]
91        133      MIT Dover Spooler                    [EBM]
93        135      Device Control Protocol              [DCT]
95        137      SUPDUP                               [15,MRC]
97        141      Datacomputer Status                  [8,JZS]
99        143      Metagram Relay                       [GEOF]
101       145      NIC Host Name Server                 [64,62,JAKE]
103       147      CSNET Mailbox Name Server (Telnet)   [58,MHS1]
105       151      CSNET Mailbox Name Server (Program)  [58,MHS1]
107       153      Remote Telnet Service                [61,JBP]
```

```
109-129   155-201   Unassigned                          [JBP]
131       203       Datacomputer                        [8,JZS]
132-223   204-337   Reserved                            [JBP]
224-241   340-361   Unassigned                          [JBP]
243       363       Survey Measurement                  [2,AV]
245       365       LINK                                [7,RDB2]
247-255   367-377   Unassigned                          [JBP]
```