Installing and Administering PPP

Edition 1

B2355-90137
HP 9000 Networking
E0497

Printed in: United States

Legal Notices

The information in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. A copy of the specific warranty terms applicable to your Hewlett- Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY 3000 Hanover Street Palo Alto, California 94304 U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

Contents

1. Introduction

Product Overview 17
Dialing In to an HP 9000 17
Dialing Out from an HP 9000 18
Direct Connections 18
HP-UX PPP Features 18

PPP and SLIP 20

Installation 21

2. Setting Up PPP Connections

Configuration Overview 24
How PPP Connections are Established 24
Configuration Quick Reference 26

Creating Device Files for Serial Ports 28

Increasing the Number of IP Tunnels 30

Configuring Your Modem 31

Configuring Outbound Connections 33
/etc/ppp/Devices 33
/etc/ppp/Dialers 33
/etc/ppp/Systems 34
/etc/ppp/Autostart 35

Configuring Inbound Connections 37
/etc/passwd 37
Login Shell Script 37
Checking Permissions 38

Testing the Connection 39

Additional Information 40
Non-Generic Login Scripts 40

Contents

Creating a Simple Filter File 40
Note on Systems and Devices Entries 41
IP Addresses on the pppd Command Line. 42

3. SLIP to PPP Migration

SLIP and PPP Configuration Files 47

Migration Examples. 50
Case 1: Dialout SLIP Connection without UUCP System Name. . . . 50
Case 2: Direct SLIP Connection without UUCP System Name. 52
Case 3: Dialout SLIP Connection with UUCP System Name 54

Setting up Outbound PPP Connections 57

Setting up Inbound PPP Connections. 58

Accounting and Logging. 59

ppl.users and ppl.ipool 60

SLIP and PPP Interoperability 61

Flow Control 62

4. Common pppd Options

Link Management 65
Active vs. Passive PPP Negotiations 65
Idle Timer Link Control 66

The exec Option 68
Parent Environment and Session Variables 68
Example. 69

SLIP Framing Option. 70
SLIP Data Compression 70
Using SLIP to Dial a Cisco Terminal Server. 70

Contents

Link Quality Monitoring 72
Guides for Evaluation of Link Quality 72
Adjusting LQM Behavior 72
Weighing the Costs of LQM 73
LQM Response to Link Failures 73
Failure Without Disconnection 73
Peer Refusal to Comply with LQM Request 74

Compression 75
HDLC Frame Compression 75
Address and Control Field Compression 75
Protocol Field Compression 75
Van Jacobson TCP Header Compression 76
PPP Link Compression 76
Predictor-1 77
Stac Compression 77

Unique PPP Implementations 78
Synchronous PPP 78
Dedicated Lines 78
Automatic Failover 79
Constantly-Open Telephone Calls 80

IP Routing Tips 81
Connecting a Host to a LAN 81
Connecting Two LANs 84
Connecting a Host or LAN to the Internet 85

5. Security Techniques

Static Packet Filtering 89
The Foundations of Security Policies 89
Filter File Rulesets 89

Filters 91

Contents

Filter Stanzas 92

Packets Overview 93
- Internet Protocol (IP) Level 93
- Internet Control Message Protocol (ICMP) Level 94
- Transmission Control Protocol (TCP) Level 95
- User Datagram Protocol (UDP) Level 96

Building a Stanza - General 97

Building a Stanza - Specifics 98
- Numbers and Addresses 98
- Keywords with Numbers 99
- Keywords with Origins and Destinations 101
- Keywords Based on TCP Packet Header Bits 102
- Keywords Based on IP Options 103
- frag Keyword 103
- all Keyword 104
- Time-Based Keywords 104
- Unreach Keyword and Sending ICMP Messages 104
- Log and Trace Keywords 106
- Stanza Syntax 107

Writing a Stanza - A Complex UDP Example 108
- An Unsafe Domain Name System Rule 108
- What Happens During Domain Name Queries 108
- Developing Safer Domain Name Request Rules 109

Writing a Stanza - TCP Examples 114
- Two Approaches to Filtering TCP connections 114
- Identifying Rulesets with Hostnames and Addresses 114
- A Note on Ruleset Formatting 115
- Ordering Stanzas Effectively 115
- Isolating an 'Incorrect' Stanza 115
- Working with Default Rulesets 116

Contents

Open Policy Default Rulesets 116
A Note on Using the 'log rejected' Filter 116
Closed Policy Default Rulesets 116
Block All Packets 117
Block All Packets Except Electronic Mail 117
Limiting Electronic Mail to a Gateway 117
Unresolvable Hostnames and Changing IP Addresses 117
Conclusion 118
A Note - Blocking Loose Source and Strict Source Routing Options 118

Closed Policy Filter Example 119
Complete Filter Example 123

Open Policy Filter Example 125
Common Mistakes 127
Complete Filter Example 128

Time-To-Call Restrictions 130

Dial-Back 131
Dial-Back Process 131
Blocking SIGHUP with Chat Script \M Option 131
Reversing Instructions with \m Option 132

Link Peer Authentication 133

Replacing getty with pppd 134

6. Troubleshooting pppd

A. Modem Connections

RS-232 Interface 145
DB-25 146
DB-9 146
DB-9 to DB-25 Conversion 147

Contents

HP Modem Cables 148
Null-Modem Cables 149

Dial Up Modems 151
Speed 151
Data Compression 151
Error Correction 152
Flow Control 153
Modem Commands 154
S Registers 155
Common Modem Configurations 156
Enabling Hardware Flow Control 159

Glossary

Printing History

The manual printing date and part number indicate its current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. the manual part number will change when extensive changes are made.

Manual updates may be issued between editions to correct errors or document product changes. To ensure that you receive the updated or new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

First Edition: April 1997

Preface

PPP is a Hewlett-Packard networking product that allows data transfer using either the Internet-standard Point-to-Point Protocol (PPP) or the non-standard, but widely-used Serial Line Internet Protocol (SLIP).

The *Installing and Administering PPP* manual provides information about configuring and using the PPP product. The manual also describes how to migrate SLIP connections from earlier HP-UX releases to the SLIP mode of the PPP product.

This manual is organized as follows:

Chapter 1	**Introduction** describes the PPP product for HP-UX.
Chapter 2	**Configuring PPP Connections** describes how to set up PPP connections.
Chapter 3	**SLIP to PPP migration** describes how to migrate existing SLIP connections to PPP connections.
Chapter 4	**Common pppd Options** describes common command line options for invoking the PPP daemon.
Chapter 5	**Security Techniques** discusses PPP's security features.
Chapter 6	**Troubleshooting pppd** describes how to solve common problems.
Appendix A	**Modem Connections** describes some basic information about modem configuration.

1 Introduction

PPP is a networking product that allows data transfer using both the Internet standard point-to-point protocol (PPP) and the non-standard but widely-used serial line internet protocol (SLIP). PPP is supplied with HP's LAN/9000 product.This chapter describes the PPP product. It contains the following sections:

- Product Overview
- PPP and SLIP
- Installation

Product Overview

Terrain, distance, and property rights often limit LAN cabling. Where coax is restricted, network connections can be made by serial lines. Serial lines are relatively inexpensive and easy to install. However, you need special software, such as PPP, to run TCP/IP applications over a serial line.

Properly installed and configured, PPP and the HP-UX system's serial port connections allow you to transmit data to remote locations through a modem or null-modem cable. (HP-UX systems are equipped with internal modems, although you can use a cable to connect an external modem or other system to a serial port.)

PPP provides an easy and flexible means of creating wide-area TCP/IP-based networks for transfering data among UNIX systems, and between UNIX systems and other systems that implement the PPP or SLIP protocols. You can use PPP to connect HP-UX systems to established TCP/IP networks. You can also use PPP to connect your HP-UX systems to a hub; this allows the HP-UX systems transparent, dial-in access to the hub and other LAN-connected hosts. Because of the transparency of TCP/IP internetwork routing, users may be unaware that remote facilities are not directly connected to the local network.

PPP includes a command program (named pppd), a startup script, and a number of sample configuration files. You can use PPP to:

- Dial-in to HP 9000 serial IP lines.
- Dial-out from HP 9000 serial IP lines.
- Direct connect to HP 9000 serial IP lines.

Dialing In to an HP 9000

Users at remote supported terminals or PCs can establish dial-in IP connections with logins. For a login connection, the user establishes a modem link to an HP 9000 and logs in as usual. After login, the Login shell script is run. The script starts pppd on the local system which will communicate with the pppd on the peer. The two pppd's will negotiate and establish a PPP connection.

For a connection without a login, the user simply dials in to a preset HP 9000 serial line where the serial protocol is already running on the line.

Dialing Out from an HP 9000

HP 9000 users can establish dial-out IP connections with or without login. This may be to any remote host that runs a supported serial IP protocol.

For a login connection, the user simply invokes pppd at the HP-UX shell prompt. pppd establishes a modem link to the specified host and logs in. pppd runs a command on the remote machine to initiate the specified serial IP protocol and establish the connection.

For a connection without login, the user's action is the same. After the user invokes pppd, a preset line is assigned to that user. The user has full network access to the remote host.

Direct Connections

You, the system administrator, may set up a direct (hardwired) IP connection between an HP 9000 and a remote host. In effect, this is simply a long distance extension of the LAN.

HP-UX PPP Features

PPP on HP-UX offers the following features:

- On demand dial-up connections. PPP can establish an asynchronous link over standard telephone lines whenever traffic warrants a call. This is in contrast to network routers that require an expensive leased line to provide a constant connection. When the traffic has completed, PPP can hang up the connection to save the telephone costs. User applications cannot distinguish between an on-demand dialup link and one that is constantly maintained, except by looking at the clock.
- Link traffic control. The HP-UX system administrator has complete control over the traffic that is allowed to traverse the link. PPP can turn an HP-UX system into a selective-isolation packet filtering "firewall" gateway, enhancing the security provided to other systems on the local network.

- Minimizes transmission overhead. PPP compresses HDLC address control and protocol fields, as well as TCP headers. This improves both throughput and interactive response on typical modem connections. The PPP interface to HP-UX TCP/IP implements Van Jacobson's TCP "fast queue" scheme, whereby interactive packets have priority to transmission on a congested link.

PPP does not support traffic encryption.

pppd is the daemon process provided with PPP to manage connections to other hosts via the point-to-point or SLIP protocols. It uses the HP-UX native serial ports. The daemon communicates with the HP-UX Kernel's TCP/IP stack via an IP network tunnel driver.

PPP and SLIP

If you must link with a peer host that cannot use the point-to-point protocol, you can use PPP's SLIP option. SLIP, a non-standard but popular protocol, is really only a framing convention for arranging IP packets on a link. Many other PPP options are available when running the SLIP option. Automatic dialing, idle line hangup, packet filtering, the exec option, and most other management facilities can be invoked in conjunction with SLIP.

However, SLIP provides few of the advanced facilities available with the PPP protocol. pppd invoked with the SLIP option *cannot* provide the following functions:

- Negotiations using link control protocol (LCP) and internet protocol control protocol (IPCP).
- Authentication using password authentication protocol (PAP) and challenge handshake authentication protocol (CHAP).
- Link quality monitoring (LQM).
- Asynchronous control character mapping or the escape option.

There are also the following connection restrictions when running PPP with the SLIP option:

- Links must be asynchronous.
- Connections must be completely transparent to all 8-bit character values.
- Flow control selection must be hardware only or no flow control at all.
- Link must not interpret passing data as flow control.

NOTE

Earlier releases of HP-UX supported SLIP and other serial line IP protocols with a facility called Point-to-Point Link (PPL), supplied with the LAN/9000 product. PPL did not support point-to-point protocol connections. If you are using SLIP to transfer data on serial IP lines, you can migrate your SLIP connections to PPP connections. Migrating SLIP to PPP connections is discussed in Chapter 3 of this manual.

Installation

You do not need to manually install the PPP software if:

- the LAN/9000 networking product was pre-installed on your system (instant ignition).
- you used the HP-UX `swinstall` program to install the Core Networking Bundle. The PPP-RUN fileset is part of this software bundle.

To determine whether PPP-RUN software has been installed on your system and is included in your kernel, do one of the following:

- Type the command:

  ```
  ls /var/adm/sw/products/Networking
  ```

  If PPP-RUN appears in the list of filesets, then the PPP software is already installed on your system.

- Type the command:

  ```
  grep tun /stand/system
  ```

  If `tun` appears in the list of filesets, then the PPP software is already included in your HP-UX Kernel.

If the PPP software is not already installed or is not included in your HP-UX Kernel, install it by running the HP-UX `swinstall` program with the command:

```
/usr/sbin/swinstall
```

Highlight the PPP-RUN filesets from the Software Selection Window and choose "Install" from the menu. When you click OK, `swinstall` will install the PPP-RUN fileset and regenerate the kernel.

The PPP-RUN fileset includes the following components:

- command program (`/usr/bin/pppd`)
- startup script (`/sbin/init.d/ppp`)
- device configuration file (`/etc/ppp/tunconf`)
- migration script for pre 10.30 HP SLIP users (`/usr/bin/ppl`)

- sample configuration files for communicating to PPP (for example, `/etc/ppp/Devices.ex`, `/etc/ppp/Dialers.ex`, and `/etc/ppp/Systems.ex`)
- example configuration files for communicating to other PPP implementations (for example, `/etc/ppp/Examples/Exec-Portmaster.ex`)

2 Setting Up PPP Connections

This chapter describes how to configure PPP for inbound and outbound connections.

Configuration Overview

A pppd process on a local system negotiates with the pppd process on a remote or peer system to establish the PPP connection. The following files are used for PPP connections:

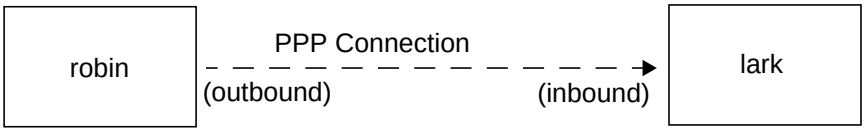
- `/etc/ppp/Autostart` starts pppd for on-demand outbound calls.
- `/etc/ppp/Systems` contains the hostname or IP address of peers and how to connect with peers.
- `/etc/ppp/Devices` associates dialer types with physical devices and speeds. pppd examines the file when it places a call. If no suitable speed is found or if all devices associated with that speed are busy, pppd tries again later.
- `/etc/ppp/Dialers` describes how to dial each type of modem attached to the HP-UX system that is to be made available for outbound calls.
- `/etc/passwd` is used to verify the login of an inbound connection.
- `/etc/ppp/Login` is a shell script that is run after a successful login. `Login` starts pppd on the local system to communicate with pppd on the peer to negotiate and establish a PPP connection.

The following section describes how these files are used to create a PPP connection.

How PPP Connections are Established

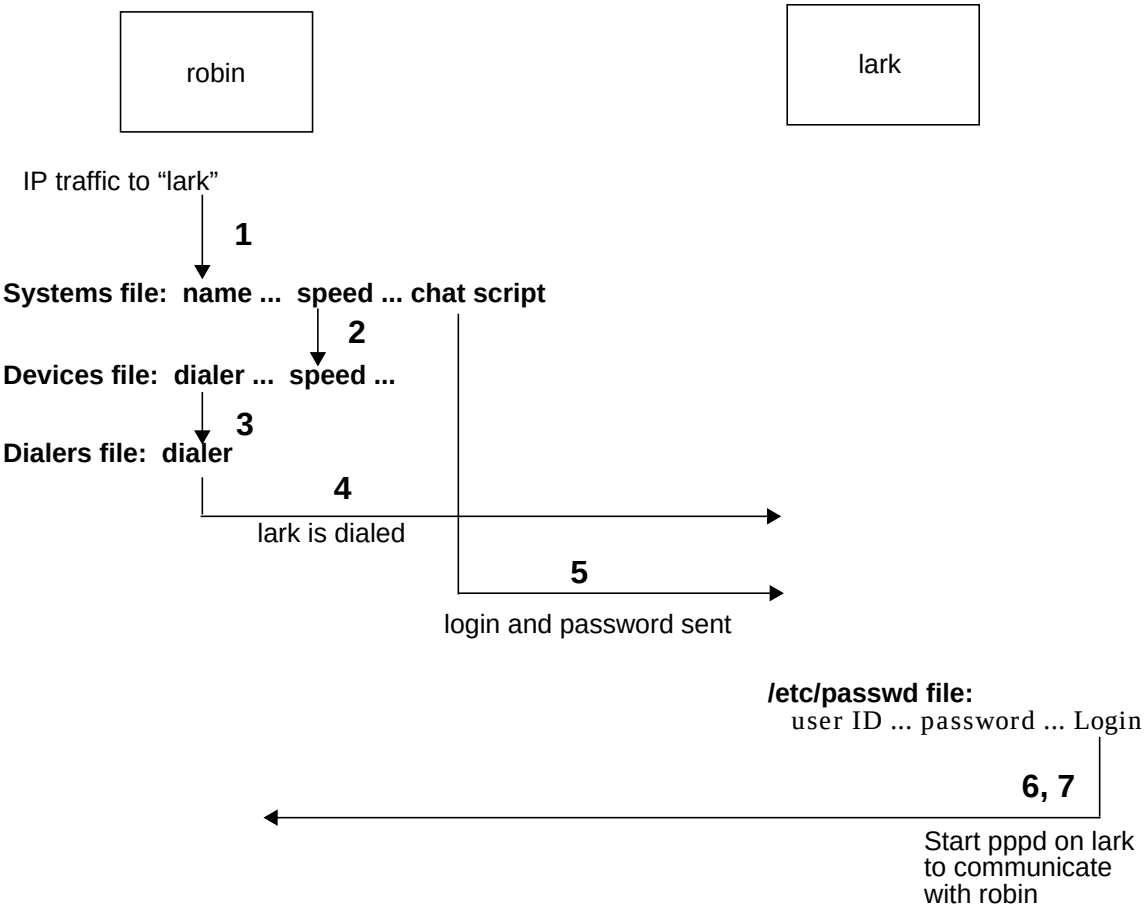
In the following example, the local system “robin” attempts to connect to the peer system “lark” via PPP, as shown in Figure 2-1.

Figure 2-1 PPP Connection Example



After PPP connections are configured on both systems and robin is re-booted, the `Autostart` file on robin starts `pppd` for an on-demand connection from robin to lark. If IP traffic is initiated to lark, the following occurs, as shown in Figure 2-2:

Figure 2-2 PPP Connection Example (continued)



1. `pppd` searches the `Systems` file on robin for an entry in the name field that matches lark's hostname or IP address.
2. When a match of lark's hostname or IP address is found in the `Systems` file, the speed field in the matching entry is noted. The `Devices` file is then searched for an entry that matches the speed field in the `Systems` file entry.

3. When a match of the speed is found in the `Devices` file, the dialer field in the matching entry is noted. The `Dialers` file is then searched for an entry that matches the dialer field in the `Devices` file entry.
4. The phone number for lark is dialed.
5. The chat script specified for "lark" in the Systems file sends robin's login name and password to lark.
6. lark verifies robin's password by comparing it to the entry in its `/etc/passwd` file.
7. If the login is successful, the Login shell script specified in the `/etc/passwd` entry is executed. The Login script starts `pppd` on lark to communicate with `pppd` on robin. The two `pppd`'s negotiate and establish a PPP connection.

Configuration Quick Reference

The following table shows the steps in configuring oubound and inbound PPP connections. Each step is described in more detail in the sections that follow the table.

Outbound	Inbound
Step 1: Create device file with dial-out or direct connect minor number for serial port. See "Creating Device Files for Serial Ports."	**Step 1**: Create device file with dial-in minor number for serial port. See "Creating Device Files for Serial Ports."
Step 2 (optional): Increase the number of IP tunnels if needed (16 are created by default). See "Increasing the Number of IP Tunnels."	**Step 2** (optional): Increase the number of IP tunnels if needed (16 are created by default). See "Increasing the Number of IP Tunnels."
Step 3: Configure your modem. See "Configuring Your Modem."	**Step 3**: Configure your modem. See "Configuring Your Modem."

Outbound	Inbound
Step 4: Create entry in `/etc/ppp/Devices`. Optionally, add entries to `/etc/ppp/Dialers` or `Dialers.local`. See "Configuring Outbound Connections."	**Step 4**: Add user accounts to `/etc/passwd` to allow incoming connections. See "Configuring Inbound Connections."
Step 5: Define dial-out connection in `/etc/ppp/Systems`. See "Configuring Outbound Connections."	**Step 5**: Create `Login` shell script. See "Configuring Inbound Connections."
Step 6: Create `/etc/ppp/Autostart`. See "Configuring Outbound Connections."	

Creating Device Files for Serial Ports

You need to create device files in the /dev directory for serial ports. Each serial port used for pppd may require one to four device files. Depending on the hardware attached, you may need files for dial-in, dial-out, or direct connection.

Use the System Administration Manager (SAM) to create device files. Creating the device files with appropriate minor numbers is important.

The following is an example of an entry in /dev for the device tty0b1, a dial-in modem with hardware flow control enabled.

```
major number = 1
minor number = 0x000012
```

The minor number corresponds to the template 0xIIPPHM, which is resolved as follows:

II Two hexadecimal digits (8 bits) to indicate the instance of the serial interface. The instance of the serial interface is determined by running the program ioscan -f and looking in the “I” column. On a series 700, port A will generally be 0 and port B will generally be 1.

PP Two hexadecimal digits (8 bits) to indicate the port number of this device on the serial interface. On a Series 700 serial port, this will always be 0.

H One hexadecimal digit (4 bits) to control diagnostic access and hardware flow control (HP J2094A only). Bit 0 controls RTS/CTS hardware flow control.

bit 0 = 0 disables the hardware flow control
bit 0 = 1 enables the hardware flow control

M One hexadecimal digit (4 bits) to determine port access type. Values for each bit are:

bit 3 = TI/ALP
bit 2 = 0 means simple protocol (U.S.A.)
bit 2 = 1 means CCITT protocol (Europe)
bits 0 and 1 = 00 direct, 01 dial-out modem, 10 dial-in modem, 11 unused

You can also use the `mknod(1M)` command to create device files, for example:

```
mknod /dev/ttyd10 c 1 0x000001 #dial-in
```

The `termio(7)` man page describes how to derive the minor number of the device file for dial-in, dial-out, and direct connections. Major and minor numbers may be different from system to system. See `termio(7)` for more details about configuring a serial device.

Increasing the Number of IP Tunnels

pppd uses IP tunnels to pass packets between the serial port and IP layer. During bootup, /sbin/init.d/ppp creates 16 IP tunnels (the default value is defined in /etc/ppp/tunconf) which will support 16 concurrent pppd processes. Each IP tunnel creates three device files. Each pppd process facilitates a PPP connection. The kernel can support a maximum of 64 IP tunnels. If you need more than 16 IP tunnels, you can increase the value of NTUN in /etc/ppp/tunconf and reboot the system.

See tun(4) for more details about configuring IP tunnels.

Configuring Your Modem

The following are general recommendations for configuring modems to use with PPP:

- Choose the highest asynchronous serial speed both modem and computer can support.
- Enable error correction. If possible, choose CCITT V.42 for compatibility with CCITT V.42bis data compression.
- Enable data compression. HP recommends CCITT V.42bis for higher maximum compression ratios and handling of precompressed data streams.
- Enable flow control. Use hardware instead of software flow control if possible.
- Use default modem parameters for purposes outside PPP protocol, including other inbound applications
- Set up dialers for UUCP or PPP outbound connections. Dialers' PPP-specific register settings ensure that a general purpose modem used for PPP will work as intended during an PPP session.
- Allow bidirectional connections over any available modem. Try the following suggestions to configure bidirectional modems and to allow modem use by other applications, such as uucp, tip, cu and remote logins:
- Set S0 register value to 1 to answer after one ring.
- Lock the DTE interface to the selected speed. Many modems do not support a speed register, but you can store the current value with the &W command
- Disable modem printing of result codes when processing incoming calls. You may have to disable result codes too. Start the dialer with ""ATE1 or its equivalent.
- Set the modem to print extended result codes, like Busy, when dialing out. Better dialer performance is possible. Enter this in the `/etc/ppp/Dialers` file, if necessary.
- Set the modem so it only asserts the Carrier Detect (CD) signal when the carrier is established with another modem

- Set the modem to disconnect the call and restore its saved values when the computer deasserts the Data Terminal Ready (DTR) signal

HP recommends that you verify that your modem connection is working correctly before configuring a PPP connection. This will detect any hardware, software, or modem configuration problems that are not related to the protocol.

To verify that the modem connection is working, use the following command to connect to a remote system.

`cu -l` *`device_file`* `-s` *`speed`*

You should be able to connect to a remote system using the `cu` command before configuring `pppd`. The remote system should display a login prompt and users can login to the remote system. This will verify that the modem line is functional. Refer to Appendix A, “Modem Connections,” for general information about modems.

Configuring Outbound Connections

The following files are used for outbound calls:

- `/etc/ppp/Devices` associates dialer types with physical devices and speeds. `pppd` examines the file when it places a call. If no suitable speed is found or if all devices associated with that speed are busy, `pppd` tries again later.
- `/etc/ppp/Dialers` describes how to dial each type of modem attached to the HP-UX system that is to be made available for outbound calls.
- `/etc/ppp/Systems` contains the hostname or IP address of peers and how to connect with peers.
- `/etc/ppp/Autostart` starts `pppd` for on-demand outbound calls.

/etc/ppp/Devices

The `Devices` file associates dialer types with physical devices and speeds. Entries in the file have the following format:

dialer device speed [*optional_parameters*]

If the PPP connection is a direct connection, the dialer field should be "Direct." Refer to the `ppp.Devices(4)` man page for more information.

In our example, the following entry is made to robin's `Devices` file:

```
T1600      cuh00      19200
```

This line provides this configuration for robin's connection to lark:

Use what modem:	T1600 (Telebit model 1600)
On which system device:	cuh00 (outbound device)
At what speed:	19200 (19200 bps)

/etc/ppp/Dialers

The `Dialers` file describes how to dial each modem that is attached to the local system. The file Dialers is installed with PPP. Entries in the file have the following format:

dialer chat_script

Refer to the `ppp.Dialers(4)` man page for more information.

In our example, the `Dialers` file is already installed on robin. Here is what the entry for the Telebit model 1600 looks like:

```
T1600   ABORT NO\sCARRIER ABORT NO\sDIALTONE ABORT BUSY \
        ABORT RRING\r\n\r\nRRING\r\n\r\nRRING ABORT ERROR \
        TIMEOUT 5 "" AT OK-AT-OK ATS111=0DT\T TIMEOUT 60 CONNECT
```

his elaborate dialer string will cause `pppd` to abort the connection attempt if anything goes wrong with the telephone call, then will disable UUCP spoofing in the modem before dialing the destination telephone number. Look through the Dialers file for modem entries for your type of modem. If none are defined, use the dialer entry "GENERIC".

You can create a separate list of modem descriptions in a file named `Dialers.local`. Entries in `Dialers.local` take precedence over entries in `Dialers`. You can use the entries from the `Dialers` file to guide you as you create new entries in `Dialers.local`. See the `ppp.Dialers(4)` man page for more information about setting up dialer entries.

Since your modem may be used for other purposes besides PPP (for example, UUCP or for interactive users), it is best to set the modem's default parameters to accommodate dial-in applications and have outgoing UUCP or PPP dialers change them if necessary. The PPP-specific register settings in `Dialers` or `Dialers.local` ensure that an otherwise general-purpose modem will work as well as possible with PPP for the duration of the PPP session.

/etc/ppp/Systems

The `Systems` file contains the hostname or IP address of peer systems and how to connect to them. Entries in the file have the following format:

name when device speed phone_number chat_script

Refer to the `ppp.Systems(4)` man page for more information.

For our example, the following line is added to the `Systems` file on robin to connect to lark:

```
lark Any ACU 19200 5551212 in:--in: Probin word: mypasswd
```

This line provides this configuration for robin's connection to lark:

Call what host:	lark
When:	Any (any day and at any time).

Using what device:	ACU (any call unit that matches the speed listed in the next field).
At what DTE speed:	19200 (19200 bps)
At what telephone #:	5551212
Expect what string:	in: substring of login: if true, send next field. If false, send string between dashes followed by carriage return and expect in: Can be used to elicit a response out of peer.
Send what string:	Probin (followed by an implicit carriage return).
Expect what string:	word (a substring of password).
Send what string:	mypasswd (followed by an implicit carriage return).

/etc/ppp/Autostart

All outbound PPP connections are started through the user-generated `/etc/ppp/Autostart` file. When the local system is booted, a `pppd` process is started for the outbound link to the remote system. There is no example `Autostart` file; you must create it. It contains the command line necessary to start `pppd`:

`pppd` *local_host*`:`*remote_host daemon_mgt_opt link_mgt_opt idle_timer*

Refer to the `pppd(1)` man page for information on command line options.

In our example, here is the `Autostart` entry on robin for connecting to lark:

```
pppd robin:lark  auto  idle  150
```

The `pppd` started here has this connection configuration:

Local and remote hosts:	robin and lark, respectively
Daemon mgmt. option:	Auto (respond to the arrival of a packet by initiating a connection to peer).
Link management option:	Idle (idle timer in effect).

Idle timer value:	150 (shut down the link if 150 seconds pass without receiving or transmitting a packet.

Configuring Inbound Connections

On machines that only accept incoming calls, pppd does not need to be started at boot time, since pppd is started when a PPP login occurs. Machines that both initiate and receive calls must start pppd at boot time, and must also prepare accounts for incoming connections. User accounts must be created in the /etc/passwd file for the system to be able to accept incoming calls.

When the local system receives a login, it does the following:

1. Verifies the password by comparing it to the entry in the /etc/passwd file.
2. If the login is successful, the Login shell script is run. Login starts pppd on the local system which will communicate with the pppd on the peer. The two pppd's will negotiate and establish a PPP connection.

/etc/passwd

The following is an example of an entry that would be made to lark's /etc/passwd file:

```
Probin::105:42:Robin's PPP login:/etc/ppp:/etc/ppp/Login
```

The '105' in the password entry is a unique user ID (uid) for this PPP login. The '42' is the group ID (gid) associated with the 'ppp' group in lark's /etc/group file.

Create a password for robin's login:

```
#  passwd Probin
New password:  some password
Retype new password:  some password
#
```

Login Shell Script

Note that a PPP user's login shell script can be located anywhere and named whatever you choose. For purposes of this illustration, the login script will be /etc/ppp/Login.

Look carefully at the last line in the sample script shown below. Notice that the word 'hostname' is surrounded by backquotes, not regular quotes or apostrophes. `hostname`, with the backquotes, tells the system to insert the output of the command `hostname(1)` in this space in the `pppd` command line. We recommend that you make sure backquotes are used by copying this script from `/etc/ppp/Login.ex`, rather than inserting them manually.

```
#!/bin/sh
# PPP login shell example
PATH=/usr/bin:/usr/etc:/etc:/bin
PPPHOME=/etc/ppp
export PATH PPPHOME
mesg  n
stty  -tostop
exec  pppd  `hostname`:robin  idle  150
```

`hostname` will return lark, the current machine, and robin is the peer. The idle timer is set to 150 seconds. Refer to the `pppd(1)` man page for `pppd` command line options.

Checking Permissions

Following the creation of the `Login` shell script, make sure the script is executable with the following command:

```
#  chmod  755 Login
```

Testing the Connection

When PPP is installed, configured, running and connected on both ends of the link, users should be able to access each peer machine using any TCP/IP application such as telnet, ftp, etc. Once you have configured the PPP connection on both the local and remote systems, follow these steps to test the outbound connection.

1. Either reboot your machine or run `/etc/ppp/Autostart` to start `pppd`.
2. Check to make sure `pppd` was started:

```
# more  /var/adm/pppd.log
8/4-14:14:43-14902  PPP
8/4-14:14:43-14902  Version 2.0
8/4-14:14:43-14902  du0:  pppd  robin:lark  auto  idle  150
```

3. Use `telnet` to bring up the link and type ^D (Ctrl-Shift-D) to exit the login. There will be a half minute pause while the local system dials the phone, the modems establish a carrier, the `Systems` chat script completes, the answering `pppd` is started on the remote system and the two `pppd`'s negotiate. For example:

```
#  telnet  lark
Trying lark...
```

Pause

```
Connected to lark.
Escape character is '^]'.

(Operating system) (lark) (ttyp3)

lark login: ^Dconnection closed by foreign host.
```

The log file will be appended to and will show how the link was initiated:

```
8/4-14:14:53-14902  tcp 137.175.2.11/1204 -> 131.187.1.131 telnet
40 syn bringup
8/4-14:14:54-14902  Dialing lark (cuh00 38400 5551212 T1600)
8/4-14:15:23-14902  PPP connected to 131.187.1.131
```

This log file snippet shows that the telnet session to lark caused the link to be initiated.

If either machine is connected to a local area network, you can set up IP routing on the two networks so that hosts on either network can communicate with hosts on the other, using machines on the ends of the PPP links as routers.

Additional Information

This section discusses some additional information that may be useful when configuring PPP connections.

Non-Generic Login Scripts

In most cases, all inbound PPP logins can use the same generic `Login` script. But if you want a host to start `pppd` with a special option like 'require authentication', make that login account use a specific login shell that is tailored to that host. Call it `/etc/ppp/Login-host`, for example, and change the `pppd` line to reflect whatever options you wish to have. For example:

```
exec  pppd  `hostname`:robin  idle  130 requireauth
```

See the `pppd(1)` man page for more information on options.

Creating a Simple Filter File

NOTE The `Filter` file is not necessary for `pppd` operation. It is only discussed here as an example on how you might use the `Filter` file.

The PPP `Filter` file specifies the ways in which static packet filtering handles outbound and inbound TCP/IP packets. Though the filtering can be very complex if desired, a simple filter will suffice for demonstration purposes between robin and lark.

An example other than the one shown below is in `/etc/ppp/Filter.ex` and a lengthy explanation of static packet filtering is included in Chapter 5.

Here is a `Filter` file that could be used for testing the robin-lark link:

```
default  bringup  !ntp  !3/icmp  !who
         keepup   !send  !ntp  !3/icmp  !who
         pass     !route
         log      !nntp  tcp/syn/recv
```

This filter defines the following:

(!ntp) (!3/icmp) (!who)

Bring up the connection for any traffic other than Network Time Protocol (NTP) packets, ICMP Network Unreachable messages, and packets from the in.rwhod daemon.

(!send) (!ntp) (!3/icmp) (!who)

Keep up the link for all packets except those sent by robin and those that will not bring up the connection.

(!route)

Pass all packets except for RIP routing messages between routed daemons.

(tcp/syn/recv) (!nntp)

Log messages when an inbound TCP session is established except for NNTP connections.

Note on Systems and Devices Entries

In both the `Systems` and `Devices` files, `pppd` selects the first line that matches its search criteria. If the connection attempt fails while using the method described by that line, `pppd` will search for the next matching line. `pppd` will report a failure only when all the criteria-matching lines have been tried and exhausted.

For example, suppose two lines in the `Systems` file differed only by the values in the telephone number field like this:

```
lark Any;50 ACU 38400 5551212 in:--in: Probin word: mypasswd
lark Any;50 ACU 38400 5551223 in:--in: Probin word: mypasswd
```

`pppd` would first try to connect by dialing 5551212. If `pppd` received a BUSY from the modem, it would dial the second number, 5551223.

Similarly, suppose a host has two different modems attached which can be used for outbound calls. The `Devices` entries might look like this:

```
T3000  cuh00  38400
USRv32bis  cul00   38400
```

`pppd` would try to call out through `/dev/cuh00`. But suppose it is busy because an incoming UUCP connection is on `/dev/ttya00`. `pppd` will try `/dev/cul00` instead.

IP Addresses on the pppd Command Line

Soft Addresses

If an IP address is input on the pppd command line, the address is offered during IPCP negotiations. However, at connection time, some terminal servers and other peers wish to assign an address for the host running PPP to use for the duration of the connection. To direct PPP to allow assignment of an address that is different from the one on the pppd command line, use a tilde (~) after the local IP address. For example:

```
pppd ‘hostname‘~:192.0.2.5 auto idle 300
```

Because SLIP does not perform any IPCP negotiations, the tilde option will not function if the SLIP option is specified. See Chapter 4 for more information.

Dynamic Address Assignment

When an answering pppd is invoked in the Login script, it is told a pair of IP addresses on the command line. In the Login script, use one of the following means to decide what IP addresses are put on the command line:

- look up the addresses in a file or a database
- calculate the addresses algorithmically based on the incoming connection's user name or other distinguishing feature
- invoke a program to ask a BOOTP server

The pppd command line arguments provide the mechanism; your Login script provides the policy.

Address Selected From a Small List

The following is an example Login script that uses the tty name to guarantee uniqueness of the addresses it assigns. This works fine for a small installation with few modem server serial ports and a fairly static configuration.

```
#!/bin/sh
TTY=‘tty‘
case $TTY in
/dev/tty1)
    IP=192.0.2.1
    ;;
/dev/tty2)
```

```
    IP=192.0.2.2
    ;;
esac
exec pppd ‘hostname‘:$IP idle 300
```

Address Calculated From tty Name

This script also uses the tty name to guarantee uniqueness of the addresses it assigns. You must define ttyN in your `/etc/hosts` file, NIS hosts map, NetInfo hosts map, or DNS database, according to the system used. This works better in a larger installation with many ports and a configuration that tends to change often.

```
#!/bin/sh
TTY=‘/bin/basename \‘/bin/tty\‘‘
exec pppd ‘hostname‘:$TTY idle 300
```

3 SLIP to PPP Migration

This chapter describes how to migrate pre-HP-UX 10.30 SLIP connections to PPP SLIP mode connections. If you did not run SLIP on a previous version of HP-UX, skip this chapter.

The purpose of this chapter is to describe how SLIP configurations may be migrated to PPP using SLIP mode on HP-UX 10.30. The SLIP-RUN fileset is replaced with a PPP-RUN fileset. The configuration requirements for PPP are similar to the SLIP configuration requirements, but the configuration files have different names and different formats.

SLIP and PPP Configuration Files

There are four main files used to set up SLIP connections with the SLIP product:

- `/etc/ppl/ppl.remotes`
- `/etc/uucp/Systems`
- `/etc/uucp/Devices`
- `/etc/uucp/Dialers`

Table 3-1 shows how the configuration information in the files and fields used with SLIP relate to the files and fields used with PPP.

Table 3-1 Relationship of SLIP and PPP Configuration Information

SLIP Files	SLIP Fields	PPP Files	PPP Fields
/etc/ppl/ppl.remotes	remote_host local_host mask protocol type uucp_name parity speed serial_line phone modem_control login_info command_name		
/etc/uucp/Systems	sysname time;retry device;protocol speed phone chat_script	/etc/ppp/Systems	name when device speed phone chat_script
/etc/uucp/Devices	device_type device_file calling_unit speed dialer_token	/etc/ppp/Devices	dialer device speed option
/etc/uucp/Dialers	modem_type chat_script	/etc/ppp/Dialers	dialer chat_script

The `/etc/ppl/ppl.remotes` file is the key to creating SLIP configuration files. The information for every dialout or direct entry may be extracted from the `/etc/ppl/ppl.remotes` file to construct the PPP configuration file. The dial-in entries in the `ppl.remotes` file have no corresponding place in the PPP configuration files.

The `/etc/uucp/Dialers` file and `/etc/ppp/Dialers` file have identical formats. The `/etc/ppp/Dialers.ex` file is delivered with PPP-RUN. To avoid potential migration issues, the `/etc/uucp/Dialers` file is copied to the `/etc/ppp/Dialers` file

during the `swinstall` process. If there is a need to move any entries from the `/etc/ppp/Dialers.ex` file to the `/etc/ppp/Dialers` file, it would be up to you, the system administrator, to do this. It is also possible for you to create a `/etc/ppp/Dialers.local` file for system-specific dialers.

For more information about PPP configuration files, refer to the following: `ppp.Dialers(4)`, `ppp.Devices(4)`, and `ppp.Systems(4)`.

Migration Examples

The sections below discuss how to set up the PPP configuration files from SLIP configuration files. Three types of SLIP configurations are used as examples.

Case 1: Dialout SLIP Connection without UUCP System Name

This type of connection is represented by an entry in `/etc/ppl/ppl.remotes` where type=DIALOUT, modem_control=YES and there is no uucp_system name. The migration task involves creating an entry in both `/etc/ppp/Systems` and `/etc/ppp/Devices` using information from `/etc/ppl/ppl.remotes` and `/etc/uucp/Devices`.

SLIP Configuration

/etc/ppl/ppl.remotes. The dialout SLIP connection without a UUCP system name is defined in `/etc/ppl/ppl.remotes` by:

keywest	# remote host name or IP address
ewok	# local hostname or IP address
	# Internet mask
SLIP	# protocol [SLIP] [ASLIPC] [ASLIPS] [PPP] [CSLIPA] [CSLIP]
DIALOUT	# type [DIRECT] [DIALIN] [DIALOUT] [DIALIN & DIALOUT]
	# UUCP system name
NONE	# line parity [EVEN] [ODD] [NONE]
9600	# line speed
cul1b1	# serial line
89659	# phone number
YES	# modem control available [YES] [NO]

login: test password: blue # log in info

“” ppl\s-i\s ewok # command name

/etc/uucp/Devices. The dialout SLIP connection is defined in `/etc/uucp/Devices` by:

```
ACU cul1b1 - 9600 hayes
```

Command Line. The command line for the dialout SLIP connection is:

```
ppl -o keywest
```

PPP Configuration

/etc/ppp/Systems. Information in the `/etc/ppp/Systems` file corresponds to the following information from the `/etc/ppl/ppl.remotes` file:

/etc/ppp/Systems	/etc/ppl/ppl.remotes
name	remote host name (keywest)
when	set to Any
device	set to ACU
speed	line speed (9600)
phone	phone number (89659)
chat_script	log in info (login: test password: blue)

After migration, the dialout PPP connection is defined in `/etc/ppp/Systems` by:

```
keywest   Any   ACU   9600   89659   in:--in: test word: blue
```

/etc/ppp/Devices. An entry in `/etc/ppp/Devices` is created by searching the `/etc/uucp/Devices` file with an entry corresponding to an entry with device_type=ACU, device_file=serial_line from `ppl.remotes`, and speed=speed from `/etc/ppl/ppl.remotes`. The `/etc/ppp/Devices` entry is created using the following information from `/etc/uucp/Devices` entry.

/etc/ppp/Devices	/etc/uucp/Devices
dialer	dialer_token (hayes)
device	device_file (cul1b1)
speed	speed (9600)

After migration, the dialout PPP connection is defined in `/etc/ppp/Devices` by:

```
hayes   cul1b1   9600
```

/etc/ppp/Autostart. The `/etc/ppp/Autostart` entry for the dialout PPP connection is:

```
pppd  ewok:keywest  auto  idle  350
```

Case 2: Direct SLIP Connection without UUCP System Name

This type of connection is represented by an entry in `/etc/ppl/ppl.remotes` by type=DIRECT, modem_control=NO and no uucp_system name. The migration task involves creating an entry in both `/etc/ppp/Systems` and `/etc/ppp/Devices` using information from `/etc/ppl/ppl.remotes` and `/etc/uucp/Devices`.

SLIP Configuration

/etc/ppl/ppl.remotes. The direct SLIP connection without a UUCP system name is defined in `/etc/ppl/ppl.remotes` by the following:

```
keywest                 # remote hostname or IP address
ewok                    # local hostname or IP address
                        # Internet mask
SLIP                    # protocol [SLIP] [ASLIPC] [ASLIPS]
                        [PPP] [CSLIPA] [CSLIP]
DIRECT                  # type [DIRECT] [DIALIN]
                        [DIALOUT] [DIALIN & DIALOUT]
                        # UUCP system name
```

```
NONE                    # line parity [EVEN] [ODD] [NONE]
57600                   # line speed
tty1p0                  # serial line
                        # phone number
NO                      # modem control available [YES][NO]
                        # log in info
                        # command name
```

/etc/uucp/Devices. The direct SLIP connection is defined in `/etc/uucp/Devices` by:

```
Direct   tty1p0   -   57600   direct
```

Command Line. The command line for the direct SLIP connection is:

```
ppl -o keywest
```

PPP Configuration

/etc/ppp/Systems. Information in the `/etc/ppp/Systems` file corresponds to the following information from the `/etc/ppl/ppl.remotes` file:

/etc/ppp/Systems	/etc/ppl/ppl.remotes
name	remote host name (keywest)
when	set to Any
device	serial line (tty1p0)
speed	line speed (57600)

After migration, the direct PPP connection is defined by:

```
keywest   Any   tty1p0   57600
```

/etc/ppp/Devices. An entry in `/etc/ppp/Devices` will be created using the information from `/etc/uucp/Devices`.

/etc/ppp/Devices	/etc/uucp/Devices
dialer	set to Direct
device	device_file (tty1p0)
speed	speed (57600)

After migration, the direct PPP connection is defined by:

```
Direct   tty1p0   57600
```

/etc/ppp/Autostart. The `/etc/ppp/Autostart` entry for the direct PPP connection is:

```
pppd ewok:keywest auto
```

Case 3: Dialout SLIP Connection with UUCP System Name

This type of connection is represented by an entry in `/etc/ppl/ppl.remotes` by type=DIALOUT, modem_control=YES and the presence of uucp_system name. The migration task involves creating an entry in both `/etc/ppp/Systems` and `/etc/ppp/Devices` using information from `/etc/ppl/ppl.remotes`, `/etc/uucp/Systems`, and `/etc/uucp/Devices`.

SLIP Configuration

/etc/ppl/ppl.remotes. The dialout SLIP connection is defined in `/etc/ppl/ppl.remotes` by the following:

```
keywest                 # remote hostname or IP address
ewok                    # local hostname or IP address
                        # Internet mask
SLIP                    # protocol [SLIP] [ASLIPC] [ASLIPS]
                        [PPP] [CSLIPA] [CSLIP]
DIALOUT                 # type [DIRECT] [DIALIN]
                        [DIALOUT] [DIALIN & DIALOUT]
kqu                     # UUCP system name
NONE                    # line parity [EVEN] [ODD] [NONE]
```

```
                                    # line speed
                                    # serial line
                                    # phone number
YES                                 # modem control available [YES][NO]
                                    # log in info
                                    # command name
```

/etc/uucp/Systems. The dialout SLIP connection is defined in `/etc/uucp/Systems` by:

```
kqu   ANY;1   ACU   9600   89659   login: test password: blueSky
```

/etc/uucp/Devices. The dialout SLIP connection is defined in `/etc/uucp/Devices` by:

```
kqu   cul1b1   -   9600   hayes
```

Command Line. The command line for the dialout SLIP connection is:

```
ppl -o keywest
```

PPP Configuration

/etc/ppp/Systems. The name for the `/etc/ppp/Systems` entry is taken from the remote host field in `/etc/ppl/ppl.remotes`. The rest of the information for the `/etc/ppp/Systems` entry is taken from the `/etc/uucp/Systems` file that corresponds to the UUCP system name defined in `/etc/ppl/ppl.remotes` (kqu):

/etc/ppp/Systems	/etc/uucp/Systems
when; retry	time; retry (set to ANY) *
device	device (must be ACU)
speed	speed (9600)
phone	phone (89659)
chat_script	chat_script (login: test password: blueSky)

* A numerical value for retry in `/etc/uucp/System` must be converted to seconds.

After migration, the dialout PPP connection is defined in `/etc/ppp/Systems` by:

```
keywest  Any;1  ACU  9600  89659  login: test password: blueSky
```

/etc/ppp/Devices. An entry in `/etc/ppp/Devices` will be created by searching the `/etc/uucp/Devices` file with an entry corresponding to an entry with device_type=ACU, and speed=speed from `/etc/uucp/Systems`. The `/etc/ppp/Devices` entry will be created using the following information from `/etc/uucp/Devices` entry.

/etc/ppp/Devices	/etc/uucp/Devices
dialer	dialer_token (hayes)
device	device_file (cul1b1)
speed	speed (9600)

After migration, the dialout PPP connection is defined in `/etc/ppp/Devices` by:

```
hayes   cul1b1   9600
```

/etc/ppp/Autostart. The `/etc/ppp/Autostart` entry for the dialout PPP connection is:

```
pppd ewok:keywest auto idle 350
```

Setting up Outbound PPP Connections

All outbound PPP connections are started through the user-generated `/etc/ppp/Autostart` file. The start-up script for PPP, `/sbin/init.d/ppp`, looks for this file and sources it if it exists. `/sbin/init.d/ppp` is shipped in PPP-RUN. The SLIP product does not have a start-up script. It is up to you, the system administrator, to create the `/etc/ppp/Autostart` file with the necessary PPP outbound connections (described in Chapter 2).

Setting up Inbound PPP Connections

On machines that only accept incoming calls, pppd does not need to be started at boot time, since pppd is started when a PPP login occurs. Machines that both initiate and receive calls must start pppd at boot time, and must also prepare accounts for incoming connections. User accounts must be created in the /etc/passwd file for the system to be able to accept incoming calls. An example entry in /etc/passwd would be:

```
pppuser::3008:2002:PPP users login (for testing):/etc/ppp:/etc/ppp/Login
```

The '3008' in the passwd entry is a unique uid for this PPP login, and the '2002' is the uid corresponding to the 'ppp' group in /etc/group.

The Login shell script, shown in the /etc/passwd example as /etc/ppp/Login, starts pppd on the local system to communicate with pppd on the peer. The two pppd's negotiate and establish the PPP connection. See Chapter 2 for more information.

Accounting and Logging

In SLIP, there is an option to create a status file which can be displayed by the command `pplstat`. There is also the ability to record connection information by creating a billing file. Similar functionality is available in PPP by using the acct and/or log options on the `pppd` command line. See `pppd(1)` for command line options.

ppl.users and ppl.ipool

In SLIP, `ppl.users` maps each user name to a single remote host. This allows a dial-in user to invoke `ppl` without identifying the remote host that is calling. It allows a dial-out user to invoke `ppl` without specifying the remote host that is being called. PPP does not provide such functionality.

The `ppl.ipool` in SLIP specifies a pool of local Internet addresses to be used for modem connections. PPP does not provide such functionality.

SLIP and PPP Interoperability

The command_name option in the `ppl.remotes` file for SLIP allows the execution of a `ppl` command on the called host. For an HP-UX system running SLIP to communicate seamlessly with an HP-UX 10.30 system running PPP in SLIP mode, either the command_name option must be changed to execute `pppd` or the 10.30 system must recognize a `ppl` command and be able to convert that to a corresponding `pppd` command. The PPP product supplies a script called `ppl` which translates the `ppl` command to a `pppd` command. It only converts inbound connections (`ppl -i`). If the user attempts to make an outbound connection (`ppl -o`) using `ppl`, an error message is issued telling the user that `pppd` must be used.

For `pppd` commands that make an outbound connection to a pre-HP-UX 10.30 SLIP system, the slip option must be used in the `pppd` command line. To start the inbound connection on the called host (SLIP host), the chat_script in `/etc/ppp/Systems` must be set up to execute a `ppl` command after login or the called host must setup execution of the `ppl` command after login (like in a `Login` script similar to what `pppd` does). It is the responsibility of the system administrator to do this setup.

Flow Control

HP ppl requires the Xon/Xoff modem setting to be turned off because SLIP and CSLIP do not allow Xon/Xoff. PPP supports the Xon/Xoff modem setting for a PPP connection. However, if pppd is run in SLIP mode, the Xon/Xoff modem setting should be turned off.

4 Common pppd Options

The PPP daemon has several sets of configuration options for fine-tuning PPP communications. The sets include over eighty different selections that affect daemon management, communications, link management, Internet Protocol, authentication, encryption, compression, logging and

signaling. pppd running on most links is basic and defined by a handful of options, but each daemon can be remarkably different with a minor change of options.

This section presents some of the more common and useful pppd command line options. Most, like the active, passive and idle timer options, control some aspect of link management. One important pppd option, filter, is discussed at length in the section on security in Chapter 5. Beyond this discussion of options for link management are short sections on synchronous PPP connections, PPP supported compression options, and IP address assignment.

Link Management

Link management options define how PPP establishes, maintains and monitors a communications link. These factors, and the condition of the link, help PPP decide when to bring a link up and down in situations other than on-demand dialups.

Active vs. Passive PPP Negotiations

HP-UX PPP, like most PPP implementations, expects to actively initiate the negotiation process. By default, when a line is connected, pppd immediately sends PPP messages, anticipating that a PPP implementation on the other end is ready to negotiate. In auto mode, this directly follows completion of a Systems chat script login procedure. When pppd is not in auto mode, messages are sent when the daemon starts.

Outbound

The default behavior works correctly in almost all situations. However, sometimes it is better to let the other end send its messages first when calling another system's dial-up modem. A few PPP implementations get confused when a peer speaks first and negotiations may be slow if both ends are active. In these situations, assign pppd the passive option. The passive option makes pppd wait until the other end communicates before it sends messages.

This is an example of the pppd command line for outbound passive connection:

```
pppd   hostname:peer   auto   passive
```

Inbound

When a PPP implementation that gets confused by the other end's active negotiations dials an HP-UX PPP system, it would be wise to make the calling pppd passive if possible. However, if it is not possible, negotiations may proceed faster if the Login script invokes the passive option on the receiving pppd.

Idle Timer Link Control

The idle option allows you to limit the number of seconds that can pass without receiving or transmitting the type of packets specified in the filter file's keepup field. The timer shuts down the link when the specified number of seconds elapse. The idle option works on both the calling and the answering pppd. If both have the idle option set, the end that specifies the shorter interval shuts down the line first. The default is to never shut down the link.

When to Invoke the Timer

Do not invoke the idle option when pppd is talking to a system that runs PPP software that does not implement on-demand auto-dialing. The session may not stay intact if the link is taken down by the idle timer. These systems include those running most free PPP implementations, or MS-DOS PCs running FTP Software's PC/TCP.

Setting Idle Timer Values

The two criteria for determining idle timer values are:

- the cost of maintaining a connection
- the scarcity of resources

Limits for Outbound Calls

Set a relatively brief idle timeout for the system placing a call if the connection costs per minute, for example, a long-distance telephone call. On the other hand, if the telephone billing scheme is based on the number of calls rather than duration, set a longer idle timer (or none at all), on the calling system's pppd. Keep in mind that it can take 25 seconds or more to allocate a modem, dial out, complete the call, login to the remote system, and negotiate PPP connectivity. The minimum idle timer setting should be 30 seconds to accommodate the connection and negotiations. All ABORT and TIMEOUT strings must be written with the same thought in mind.

Limits for Inbound Calls

Set a relatively brief timeout for an answering system if it has too few modems to accommodate a large number of incoming connections. The answering system might also benefit from a shorter idle timeout value if it acts as a hub and provides its services via a toll-free WATS (800)

number. On the other hand, if it provides its services via a 976 or 900 number scheme, you may not want to specify any timeout interval since each unit of connection time increases the answering system's revenue.

The exec Option

The exec option invokes a command or shell script when a link is brought up or taken down. Exec can be used on the link's calling or answering end and the command or shell script is executed immediately when the link changes state.

The first argument of the command or shell script invoked by exec is either “up” or “down." The second argument is the peer's IP address. Exec's third and subsequent arguments are those with which pppd was invoked.

NOTE

Note: pppd runs suid root and anything specified in the Exec script is run as root. This can be a risk if mismanaged. Take appropriate security precautions with the exec script so it cannot be modified by non-root users.

Parent Environment and Session Variables

The entire parent environment is exported to exec scripts (the environment when pppd was invoked.) You should attempt to make the environment as small as possible because the environment will take up space for each pppd that is running. The exec script may also include variables that describe the current session. These variables include:

Variable	Description
PPPHOME	As you would expect, but always set.
PPP_COMMANDLINE	All arguments from the original command line.
PPP_LOCALADDR	Local address of link while up (negotiated).
PPP_REMOTEADDR	Remote address while up.
PPP_NETMASK	Netmask while up.
PPP_ORIG_LOCALADDR	Original local address (from command line).
PPP_ORIG_REMOTEADDR	Original remote address.

Variable	Description
PPP_ORIG_NETMASK	Original netmask.
PPP_INTERFACE	Name of PPP interface (DUX).
PPP_DEVICE	Name of device being used for serial stream.
PPP_LOGFILE	Self-explanatory.
PPP_ACCTFILE	Self-explanatory.
PPP_AUTHNAME	Peer name if CHAP/PAP is used, else “name” parameter from command line, if given. Else user name of user that started pppd.
PPP_PID	pppd user process PID.
PPP_SHUTDOWN	Shutdown reason if link is going down and a shutdown reason is available, else non-existent.

Example

This example uses a `Login` shell script to invoke an executable shell script called `$PPPHOME/Exec`. `$PPPHOME/Exec` causes `sendmail` to run its queue whenever a remote system establishes a connection, possibly triggering a delivery to the remote system.

Login Shell Script

```
#!/bin/sh
PATH=/usr/bin:/usr/etc:/etc:/bin
PPPHOME=/etc/ppp
export PPPHOME PATH

mesg n
stty -tostop
exec   pppd   ‘hostname‘:   idle   130   exec $PPPHOME/Exec
```

Executable Shell Script:

```
#!/bin/sh
case "$1" in
    up)     echo   link is up at ‘date‘ >   /dev/console ;
            /usr/lib/sendmail -q < /dev/null ;;
    down)   echo   link is down at ‘date‘ >   /dev/console ;;
esac
```

SLIP Framing Option

Unlike PPP, SLIP cannot perform Internet Protocol Control Protocol (IPCP) negotiations. Therefore a SLIP connection cannot be assigned an address by a peer at connection time. Normally, PPP can use the tilde (~) on the pppd command line for this purpose. To obtain a similar "feature," a remote side SLIP connection must provide the IP address during the connection process and a new local address must be obtained using the Systems file `\A` chat script feature.

The Login script can tell the peer its address textually, just before invoking pppd, if you include something like this in the script:

```
#!/bin/sh
localaddr='calculate'
peeraddr='calculate'
echo my address is $localaddr and your address is $peeraddr
exec pppd $localaddr:$peeraddr slip idle 120 ...
```

The incoming peer is responsible for parsing the "my address is" line, and doing something sensible with the addresses it finds there.

SLIP Data Compression

If you specify vjcomp and SLIP as pppd options, pppd will always use RFC 1144 'VJ' TCP header compression with the default 16 slots. This is useful for talking to 'CSLIP', a SLIP protocol that allows compression. By default, pppd running with the SLIP option does not use VJ header compression, although it will start sending VJ-compressed packets if it first receives VJ-compressed messages from its peer. Header compression can be completely disabled with the novjcomp option. SLIP's default maximum receive unit (MRU) and maximum transmission unit (MTU) values are 1006.

Using SLIP to Dial a Cisco Terminal Server

You can use PPP in its SLIP framing mode to dial into a Cisco systems terminal server. Since the SLIP protocol supports no option negotiations, Cisco terminal servers print the incoming system's IP address in text on the serial line, before beginning the SLIP protocol. pppd parses when the '\A' token is inserted in the 'expect' phase of the Systems file chat

script. The daemon passes the assigned address to the UNIX end of the point-to-point networking interface. See `Systems.ex` for an example use of this facility.

Link Quality Monitoring

The Link Quality Monitoring (LQM) option is defined in the PPP protocol specification. When the PPP implementation supports LQM, PPP can make policy decisions based on the quality of the link between peers.

When LQM is invoked, pppd requests that the other end of the connection send Link Quality Report (LQR) packets back to pppd. If the link goes down or is degraded, many or all of the LQR packets will be lost. This also indicates that data packets have been lost and the connection is too bad to warrant continued traffic. pppd counts the arriving LQRs, and if too many have been lost, pppd closes the connection. LQM packets do not count against the idle line disconnection timer, since they are part of the protocol rather than user data.

Guides for Evaluation of Link Quality

pppd bases its evaluation of the link status on the arguments of two associated options, lqrinterval and lqthreshold. The value of the lqrinterval option tells pppd how often to request LQR packets from the peer machine. The default value is every ten seconds, or five per minute, allowing for timing slop. lqthreshold's argument is the minimum acceptable link quality. The default lqthreshold is one out of five packets. With the default values in place, the link will shut down if no LQR packets arrived during the previous minute.

Adjusting LQM Behavior

The default LQM behavior, one successful packet out of five sent every minute, is fairly permissive. pppd is more demanding about line quality if you change the defaults. Either evaluation setting can be changed to make LQM more stringent. For example, you can demand that at least one LQR arrive per minute by specifying

```
lqrinterval 5 lqthreshold 1/12
```

Or you can demand that no more than one LQR be lost each minute by specifying

```
lqrinterval 5 lqthreshold 11/12
```

Weighing the Costs of LQM

pppd discovers line failures more quickly if you decrease the lqrinterval because LQR's will arrive more frequently. However, sending more LQR's increases monitoring traffic, slowing down the transfer of user data. So you should remember to also consider raising the lqthreshold. If the lqthreshold is 'lqthreshold 5/6', no more than one LQR can be dropped per minute. If two LQRs in a row are dropped, pppd shuts down the line.

LQM Response to Link Failures

LQM is particularly useful for detecting and responding to total link failures. The response can include failover protection which switches the connection to an available dial-up modem. Most total failures, for example, a backhoe digging through the telephone company's cable, cause the connection's CSU/DSU or modem to deassert the Carrier Detect signal. pppd observes this as a hangup event.

Failure Without Disconnection

But some failure modes, like misconfigured flow control and over-reliance on in-band XON/XOFF flow control, leave modems connected even though the PPP peers are unable to communicate. In this situation, the peers observe an LQM failure and take appropriate action, usually disconnecting and redialing. LQM is also useful if an intercontinental telephone connection is of such poor quality that significant numbers of packets are damaged in transit. Again, the PPP implementations can decide, based on LQM measurements, to hang up and redial in hopes of getting a better connection. This is a very rare occurrence if your modems provide V. 42 or MNP 4 error correction, but it does occasionally happen. Whatever the cause, the disconnection and redial operation can happen without user intervention or application awareness. This is because even if PPP frames are damaged or lost, the upper protocol layers arrange for retransmission as needed to provide the user with a complete data stream. The user simply experiences a pause while the modems reestablish the connection.

Peer Refusal to Comply with LQM Request

If, during LCP option negotiation, the peer refuses to send Link Quality Reports, pppd instead begins sending LCP Echo-Request messages at the requested lqrinterval and use the arriving LCP Echo-Response messages to make the link quality decision. If the peer does not correctly use a LCP Configure-Reject message to tell PPP to switch to LCP Echo-Requests, PPP can be given either the echolqm argument, to dispense with the negotiation phase and begin directly with Echo-Requests, or the nolqm argument, to disable link quality monitoring completely. At pppd's debugging verbosity level 4, the log file receives summary messages like this:

```
5/7-13:45:27-1514 LQM: Pkt: 1/1 Oct: 53/53 LQRs: 5/5
```

This means that, during the last testing interval, this system transmitted one packet and received one packet. Fifty-three octets crossed the link in each direction. And this system has received responses to all five of the most recent five Link Quality Reports it sent. The LQR packet is 36 octets long, and the default lqrinterval of ten seconds will cause the additional traffic to be unnoticed on most connections. However, if the application is very sensitive to speed and requires absolutely every bit of available line bandwidth, use the nolqm argument to disable LQM.

Compression

In addition to in-modem data compression, PPP supports compression at several different layers of the communications stack.

HDLC Frame Compression

The PPP frame format is based on the established HDLC format. Synchronous PPP links almost always use the full PPP/HDLC frame because the link hardware supports it. But lower-speed asynchronous links typically handle framing in software and several of the fields carry the same contents in each message. Therefore, it makes sense to amend the full HDLC frame for asynchronous PPP.

Address and Control Field Compression

A full PPP frame contains the HDLC ALLSTATIONS value (0xFF) in the address field, and the HDLC Un-numbered information value (0x03) in the control field. These fields are unnecessary because they always carry the same information and increase the latency of a low-speed link. During the Link Control Protocol (LCP) phase, asynchronous PPP links usually negotiate to drop both fields. When the LCP layer is opened, the fields are not transmitted in PPP frames.

Protocol Field Compression

The two-octet PPP Protocol field in an asynchronous PPP frame tells the receiving PPP whether the incoming frame carries an IP network datagram, a Link Quality Report, a link option (re)negotiation, or any of several other types of data. Though option negotiations and other link-level traffic always require two octets, most of an established link's traffic is network-layer (for example, IP) datagrams. PPP's network-layer protocol field values always place a null octet (0x00) before the octet that distinguishes IP (0x21) from Appletalk (0x29) datagrams. That octet almost always contains the same value (0x00), and asynchronous PPP links usually negotiate it away to reduce latency.

Van Jacobson TCP Header Compression

Each layer a TCP/IP datagram passes through adds a header to the user data. For example, many streams can potentially pass between two hosts. Therefore, in addition to its source and destination addresses, each packet contains a tag to identify which stream it belongs to. These headers are very large, and a comparison between successive packets reveals strong similarities.

RFC 1144 'VJ' TCP header compression reduces the packet header size by transmitting only the header segments that change from one packet to the next. TCP and IP header overhead is reduced from over 40 octets to as few as 4. TCP header compression has a dramatic effect on interactive responsiveness over low-speed links, because it reduces a typical single-character Telnet or rlogin packet from over 40 octets to 5 or 6 octets. It has a much smaller effect on batch data throughput, like X bitmap displays, or FTP or rcp file transfers. That sort of data flows in much larger packets. Reducing frame size from 1500 to 1460 octets produces a much smaller percentage improvement than a reduction from 45 to 5 octets.

The option vjslots followed by a numerical value between 3 and 256 sets the number of compression slots for Van Jacobson compression. The default is 16 slots. See the pppd man pages for other options for VJ compression.

TCP header compression is enabled by default on asynchronous PPP and SLIP links, and disabled by default on synchronous PPP links.

PPP Link Compression

Like in-modem data compression, PPP link compression reduces the amount of data that must flow across a low-bandwidth telephone line, thus increasing its effective bandwidth. Since PPP link compression in pppd is performed on the UNIX system, less data flows across the serial interface. This is advantageous in the following situations:

- The host's serial interfaces are incapable of high asynchronous data rates.
- The host's inefficient serial I/O subsystem causes an onerous interrupt load on the host processor.
- Two computers are directly connected and there are no modems to compress the data.

- The modem or CSU/DSU communication device does not support data compression.

PPP link compression over modems that support V.42bis compression may provide no performance advantage, except in cases where the link's bandwidth is limited by slow serial interfaces.

Predictor-1

Predictor-1 compresses typical binary data 1.5:1, absorbs relatively little of the host's CPU, and adds very little latency to interactive traffic. It is well suited to achieving even better performance from higher-speed synchronous PPP connections.

Stac Compression

The Stac LZS algorithm for data compression is an Internet Engineering Task Force (IETF) compression standard for PPP. The algorithm can maintain an average 2:1 compression ratio and, depending on the type of data that is being compressed, the ratio may be as high as 3:1 or 5:1 . The Stac LZS algorithm is based on the Lempel-Ziv algorithm and is the same as that used in Stac Electronics retail software.

Unique PPP Implementations

Although most implementations of PPP occur over aysnchronous dial up connections, PPP can be used for synchronous transmission over high speed serial interfaces. It can also be used on dedicated lines and constantly open telephone lines. The latter is a dial-up connection, but it is not on-demand.

Synchronous PPP

PPP can run in synchronous mode using a high speed serial interface at line speeds up to T1 (1.544Mb/s). To prepare your system to use a high speed interface, follow the instructions in the hardware installation guide.

Dedicated Lines

Use pppd's dedicated argument if the PPP implementation uses asynchronous serial connections that are always available. These connections often use high-speed asynchronous short-haul modems over a building or campus wiring plant. The dedicated argument instructs pppd to never give up on the connection. If the peer tells pppd to disconnect, pppd will continuously attempt to reconnect and connectivity is reestablished as soon as possible if one end of the link goes down.

If there is a fatal disconnect, through LQM failures or loss of the Carrier Detect signal, pppd closes the device and consults the Systems file to find another matching entry. If none is available, pppd waits for the call retry delay to pass and tries the original connection again. Normally, no getty or login process is run on a dedicated line device and both ends of the circuit actively try to connect to their peer. Each machine's Autostart script should contain a line like the following:

```
pppd local:remote auto dedicated
```

The Systems file should specify a device name like cuh00 in the device field. ACU, for "Any Call Unit", should not be used. For example:

```
remote Any cuh00 38400 0
```

The Devices file should contain a line like the following:

```
Direct cuh00 38400
```

The `Dialers` file is ignored when "Direct" is found in the dialer field of the `Devices` field. See the discussion below regarding line failovers and using an auto-dial modem as a backup link.

Automatic Failover

The Automatic Failover option is a dial-up backup that maintains connectivity so that IP traffic can continue when a synchronous or dedicated asynchronous connection is dropped. User services continue even if the dedicated line becomes unavailable, although the user may notice the link is slower.

Setting up Automatic Failover

To set up the dial-up connection, add an entry referencing a dialup modem after the entry for the dedicated link in the `Systems` file. The added line might look like the following:

```
remote Any ACU 19200 5551212 in:--in: pppbackup word: password
```

The remote hostname must match the remote hostname entered in the `Autostart` file entry described earlier in the section "Dedicated Lines." You also need an entry in the `Devices` file that accesses a device the modem is attached to. Instead of using "Direct" in the dialer field, substitute a dialer entry for your modem. For example:

```
USR-SPORTSTER cuh00 19200
```

In this example, the `Dialers` file would have a "USR-SPORTSTER" dialer entry to use to dial out. The modem would be attached to a serial port which is accessed through device "cuh00" with a DTE speed of 19200.

How Automatic Failover Works

By default, `pppd` asks the peer to send LCP LinkQualityReport messages. When the dedicated line fails, `pppd` stops receiving the reports. `pppd` terminates the connection when the lack of Link Quality Reports drives measured link quality below the configured threshold. After unsuccessfully attempting to reestablish the connection on the same line, `pppd` automatically fails over to the second entry in the `Systems` file, and uses the modem to dial up and reestablish IP traffic. Note that if the dedicated connection is restored, you must manually cause the dialup modem to hang up the line. Then `pppd` attempts to

reconnect using the next entry in Systems, or, if no additional entries exist, pppd wraps around to the first entry in the file which is the dedicated connection.

Constantly-Open Telephone Calls

Some PPP connections are always up. The system does not use pppd's on-demand dialing to reestablish a link for new traffic. This is not the same as using a dedicated line, because modems on constantly open connections must be dialed, or a login negotiated, before PPP frames can be exchanged. In the "constantly up" situation, use the "up" argument on pppd's command line. The argument "up" instructs pppd to make every effort to keep the connection up. For example, when the connection goes down, pppd immediately redials the modem, rather than waiting for traffic demand. Do not use the up and idle arguments together.

IP Routing Tips

The UNIX host's IP implementation sees PPP as a point-to-point network connection between two known addresses. If neither end-point resides on an IP-based local area network (LAN), packets simply flow in both directions as soon as the PPP connection is established. When the PPP link connects a remote host to a LAN, provides a connection between two LANs, or connects a host or a LAN to the worldwide Internet, the systems involved must be concerned with routing.

The examples below describe how to establish static routes when the machines boot. This is cumbersome, because any topology change requires that all the machines even remotely involved be reconfigured by editing their boot-time shell scripts. An alternative to static routes would be to use some automated system for updating all the hosts' routing tables. This is usually implemented as a daemon running on all the hosts involved. Many networks use the RIP protocol and run `routed` on their UNIX systems. If you use `routed` to propagate routing information, you will need to specify the PPP link as leading to a passive gateway.

Some sites prefer other routing protocols, and they use `gated` instead. If you use `gated`, you should invoke `pppd` with the netmask argument set to the same value as used on the LAN, if that subnet mask is different from the default for the class of your network.

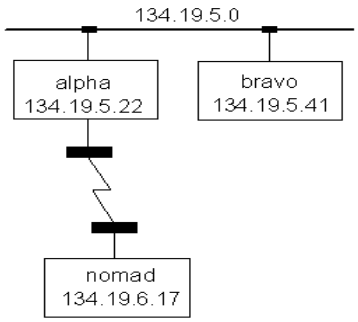
Connecting a Host to a LAN

There are two conventional approaches for connecting a set of standalone hosts to a LAN via PPP.

Separate Network

The method that causes the least confusion is to assign a network or subnet for use by all remote machines. For example, suppose the organization's class B network number is 134.19, and they are subnetting with a class C-sized network mask of 0xffffff00. The departmental LAN is subnet 134.19.5.0, populated by hosts alpha (134.19.5.22) and bravo (134.19.5.41). A modem is attached to one of alpha's serial ports, and alpha's `Login` shell script is a generic one as shown below. A laptop named nomad wishes to connect to the network.

Figure 4-1




```
#!/bin/sh
PATH=/usr/bin:/usr/etc:/etc:/bin
mesg n
stty -tostop
exec pppd `hostname`: idle 180
```

Designate subnet 6 for remote machines. Assign nomad the IP address 134.19.6.17. The PPP daemon on nomad would be started in the `Autostart` script as:

```
pppd 134.19.6.17:134.19.5.22 auto idle 180
```

or, if all the names can be found in the local `/etc/hosts` (or resolved via NIS/YP, NetInfo, the Domain Name Service, or some other mechanism without first needing to bring up the PPP link), it could look like

```
pppd nomad:alpha auto idle 180
```

The next line in nomad's `Autostart` would set up an IP route through the dial-in gateway:

```
route add net 134.19.0.0 134.19.5.22 1
```

Alternatively, and necessarily if the LAN were also connected to the Internet:

```
route add default alpha 1
```

Similarly, the `pppd` on alpha would be started as

```
pppd alpha:nomad auto idle 180
```

bravo needs to have a network route for 134.19.6.0 pointing through alpha.

```
route add net 134.19.6.0 alpha 1
```

ARP Table Manipulation. If a separate subnet number is unavailable for use by remote-access machines, it is possible to assign the remote machines addresses on the same subnet number as the departmental LAN. As above, suppose the organization's class B network number is 134.19, and they are subnetting with a class C-sized network mask of 0xffffff00. The departmental LAN is subnet 134.19.5.0, populated by hosts alpha (134.19.5.22) and bravo (134.19.5.41). A modem is attached to one of alpha's serial ports, and alpha's `Login` shell script is the generic one described above. A laptop named nomad wishes to connect to the network.

Assign nomad the IP address 134.19.5.114. The PPP daemon on nomad would be started in the `Autostart` script as

```
pppd 134.19.5.114:134.19.5.22 auto idle 180
```

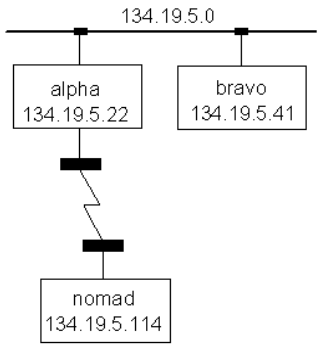
or, if all the names can be found in the local /etc/hosts:

```
pppd nomad:alpha auto idle 180
```

The next line in nomad's `Autostart` would set up an IP route through the dial-in gateway:

```
route add net 134.19.0.0 134.19.5.22 1
```

Figure 4-2



Alternatively, and necessarily if the LAN were also connected to the Internet:

```
route add default alpha 1
```

Similarly, the `pppd` on alpha would be started as:

```
pppd alpha:nomad auto idle 180
```

Alpha would run the following command at boot time:

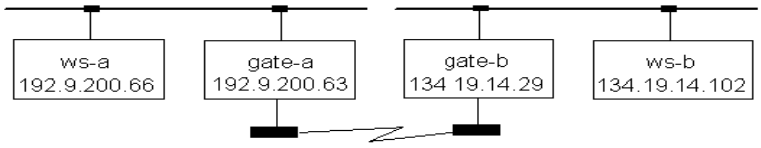
```
arp -s nomad 8:0:9:30:dc:91 pub
```

(The hexadecimal sequence 8:0:9:30:dc:91 is the Mac address of the Ethernet card in alpha. Substitute the Mac address for your machine's Ethernet card in the command above.) This would add a permanent entry to alpha's ARP table, and cause it to be provided to other systems on the local Ethernet. Any time a host on that LAN tries to find the Ethernet address corresponding to nomad's IP address, the request will be answered with instructions to forward the packets to alpha. Since nomad appears to be directly connected to the LAN, no hosts on the LAN, nor on any other IP-connected network, would require routing table modifications.

Connecting Two LANs

Suppose gate-a (192.9.200.63) and ws-a (192.9.200.66) are on one LAN, with gate-a supporting a modem. Also suppose gate-b (134.19.14.29) and ws-b (134.19.14.102) are on another LAN across town, with gate-b supporting a modem.

Figure 4-3



gate-b's pppd should be started as

```
pppd gate-b:gate-a auto idle 130
```

and gate-b should have a route like

```
route add net 192.9.200.0 gate-a 1
```

ws-b should have a route like

```
route add net 192.9.200.0 gate-b 1
```

Similarly, gate-a's pppd should be started as

```
pppd gate-a:gate-b auto idle 130
```

and gate-a should have a route like

```
route add net 134.19.14.0 gate-b 1
```

or, depending upon the structure of LAN b, maybe

```
route add net 134.19.0.0 gate-b 1
```

ws-a should have a route like

```
route add net 134.19.14.0 gate-a 1
```

or, again depending upon the structure of LAN b, perhaps

```
route add net 134.19.0.0 gate-a 1
```

Connecting a Host or LAN to the Internet

If your LAN is connected to the Internet, or if you have arranged an account at a Point Of Presence (POP) of a PPP or SLIP-talking Internet connectivity vendor (say foo.net), then you should arrange for your default route to point through the gateway at the other end of the PPP connection. If hotel supported a modem to call a POP, it would start its PPP daemon like

```
pppd hotel:pop.foo.net auto idle 240
```

and would arrange a route as

```
route add default pop.foo.net 1
```

Any hosts on the LAN 'behind' hotel would set a route like

```
route add default hotel 1
```

A machine's default route should point to the next machine along its path "outward" to the Internet. If hotel were a remote machine dialing into one machine in a complex corporate internet, its default route should point to that hub machine, expecting that the hub will deal with the issues of routing hotel's packets to their destination, whether on the LAN or elsewhere on the corporate internet or onto the Internet.

5 Security Techniques

It is impractical to impose thorough security policies on each internal host of the networks linked by an PPP connection. But PPP's strong security features support a variety of techniques that strengthen your network's ability to prevent loss.

In most cases, a single connection can be supported by more than one of PPP's security features. For example, a connection might use any of the following:

- Packet Filtering
- Time to Call Restrictions
- Dial Back
- CHAP Authentication

Static Packet Filtering

We recommend that you establish a security policy before you write a packet filter. A security policy is a statement based on thorough analysis of access needs, vulnerabilities, and real, or perceived, threats to your assets. You must identify the types of network traffic associated with these issues before you can create a packet filter that supports your security policy.

The Foundations of Security Policies

In general, all security policies are based on one of two opposing strategies. Both types of policies are supported by PPP filters.

The first strategy permits a few specific services and blocks everything else. If you follow this philosophy, a service will be unavailable if you commit an error of omission. This is a fail-safe, or closed, policy.

The second strategy blocks only specific services and permits everything else. If you begin from this premise, an error of omission may leave you unintentionally vulnerable when a fragile service is not blocked.

If you need aid in developing security policies, or would like more general information about network security and packet filtering, you should begin by reading two books, *Firewalls and Internet Security* by Bill Cheswick and Steve Bellovin and *Building Internet Firewalls* by Brent Chapman and Elizabeth Zwicky.

Filter File Rulesets

When pppd starts, the software checks for a filter file. If one is present, it is parsed and installed. The default filter filename for pppd is `Filter`. If you want to give the file a different name, specify the new name as the argument for the pppd 'filter' option. Only add the filter option if you want to change the name of the filter file.

A filter file contains rulesets for filtering packets. Each ruleset begins with one of the following:

- an IP address
- a hostname

- the special keyword, ‘default’.

You may write a specific ruleset for each connecting host, or a default ruleset will be used. The pppd parser searches for a ruleset that matches the IP address or hostname of the remote PPP/SLIP host, called the peer. This usually corresponds to the IP address placed on the right hand side of the colon on the pppd command line.

Ruleset Design

Rulesets are designed on a per-connecting-host basis rather than a per-interface basis. This provides support for devices acting as PPP or SLIP routing hubs. A hub workstation allows multiple hosts to establish IP connections and may support multiple hosts establishing connections at different times on the same interface.

If a hub supports different classes of users, PPP filters allow you to define different access policies for each group. A single hub workstation may support all of the following PPP/SLIP connections, each defined by a different ruleset:

- a connection to the home of a developer who is allowed to access multiple hosts and proprietary data
- connections for members of a traveling sales team who only require electronic mail access
- connections to customers seeking support who may only access the anonymous FTP host

Default rulesets are permitted. They simplify configuration when a single machine supports similar multiple hosts/connections that can be controlled by the same security policy.

Ruleset Order

The order in which rulesets appear is important. Default rulesets should appear early in the file because, after they are parsed, the parser continues searching for more matching rulesets. However, when addresses or hostnames in packets and rulesets match, the packet is processed and the parser stops its search.

When a match is found and the ‘non-default’ ruleset is processed, individual filters replace any default filters "remembered" from earlier in the file. This means that packet filtering may behave differently if the "default" rule appears early or late in the file.

Filters

A ruleset is made up of one to four filters that regulate the response to a packet. The filter's actions are defined by its initial keyword. Each type of filter may be used one time per connection. The following table explains the keywords, the types of packets affected by the filters, and the filter's actions:

Keyword	Packet Type	Action
bringup	outgoing dialup	Defines packets that cause a connection to be established.
keepup	inbound and outbound dialup	Defines packets that cause the idle timer to be reset, preventing the connection from going down.
pass	all packets	Defines packets that are allowed to pass through the filter. Packets that do not pass cannot cause a connection to be established.
log	all packets	Defines the characteristics that will cause a message about a packet to be added to a log file.

Filters defined in a ruleset replace any previous default definition for that filter. Defined filters are not additive with a default filter. If one of the keyword filters does not appear in a ruleset, that filter is defined by its the most recently parsed default ruleset. If there is no previous default ruleset, the implicit default is 'all', except for the log filter, which defaults to '!all'.

Filter Stanzas

Each filter is composed of a filter name followed by one or more stanzas (rules). Each packet passing through the interface is compared to the rules in the stanzas until a match is found, completing the filter operation. The ordering of the stanzas is therefore extremely important. Packets may match on many types of values, including:

- host or network address
- port number
- protocol type
- IP option
- TCP SYN, FIN, ACK and RST bits
- direction of traffic

A stanza optionally begins with the negation operator, the exclamation mark (!). The mark is followed by one or more values and keywords, each separated by a slash (/). These value and keyword combinations create a specification for a packet.

An exclamation mark is a powerful specification in any filter rule. If an exclamation mark is placed at the beginning of a rule, it negates the action of the stanza's keyword. For example, compare the defaults for the Pass and Log filters:

```
Pass
      all
Log
      !all
```

The specification for Pass, and the lack of an exclamation mark, indicates the default is to pass all packets. On the other hand, the '!all' default designation for Log means no packets are logged.

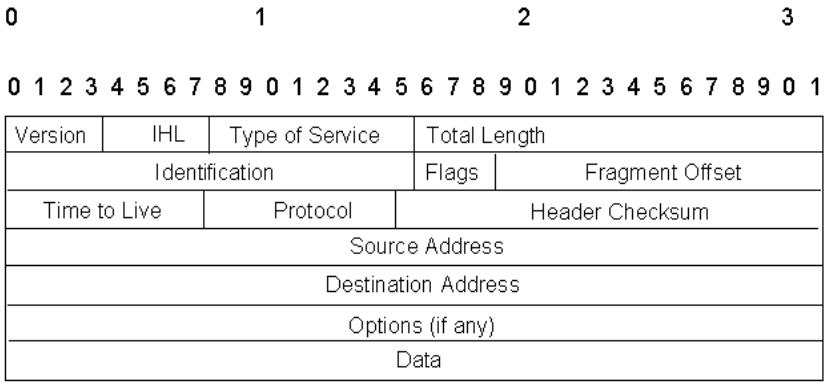
Each filter ends with an implicit stanza. The implicit ending stanza is 'all' if the last stanza specified is negated by beginning with '!'. The implicit ending stanza is '!all' if the last stanza is not negated. While this is convenient and makes it unnecessary to actually define the final "match everything else" stanza, it is a good idea to explicitly specify this stanza to avoid simple errors that can greatly change the meaning of a filter.

Packets Overview

Each stanza represents a template used to find matching packets. The features you can place in your stanzas correspond to the fields in network messages. This allows you to filter at the IP level or the TCP/UDP/ICMP level.

Internet Protocol (IP) Level

Figure 5-1



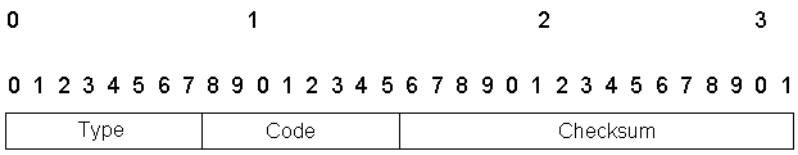
*RFC-791 (IP)

The fields available at the IP level include the protocol (e.g., 1 [ICMP], 2 [IGMP], 6 [TCP], 8 [EGP], 17 [UDP], etc.), an address (i.e., the source or destination address) and IP options (e.g., rr, lsrr, etc.), and fragmentation.

PPP does not provide a filter keyword for matching the version, IHL, TOS, length, ID, TTL or checksum header fields.

Internet Control Message Protocol (ICMP) Level

Figure 5-2



* RFC-792 [ICMP]

ICMP messages may be filtered on the type and code fields.

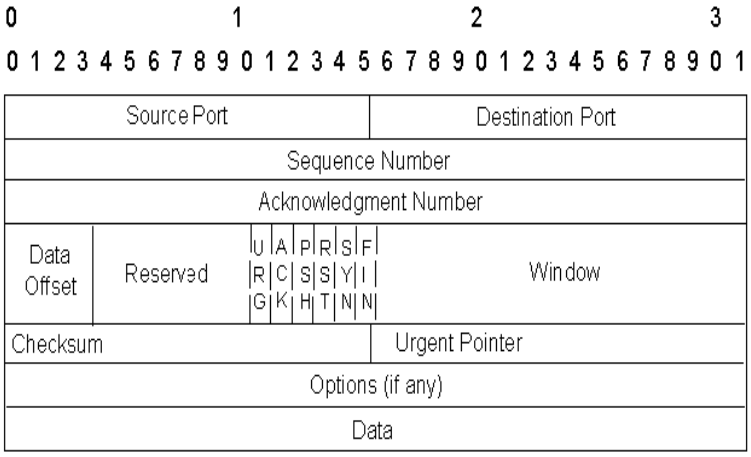
In general, it is not a good idea to block all inbound or outbound ICMP messages because ICMP messages are an important way that status information is conveyed over an IP network. For instance, blocking ICMP Source Quench messages (Type 4), used to tell a packet source to slow down, can cause problems for other users and sites.

It is true that you should probably not permit ICMP Redirect messages (Type 5) to pass through your router since the routing on an internal node should not be changed by an external site.

If you want to block ping from being used for host discovery, then you should block inbound ICMP Echo packets (Type 8).

Transmission Control Protocol (TCP) Level

Figure 5-3



* RFC-793 [TCP]

The TCP header fields available for matching include port numbers for the source or destination port, the presence of the SYN bit without ACK, and the ACK, FIN and RST bits.

PPP does not provide a method for filtering on TCP options, the presence of URG/EOM bits in the TCP options, or other TCP header fields.

Establishing a TCP connection requires synchronization. Each side must send its own initial sequence number, receive a confirming acknowledgment from the other end, receive the other end's initial sequence number and send the confirming acknowledgment.

The steps in the sequence look like this:

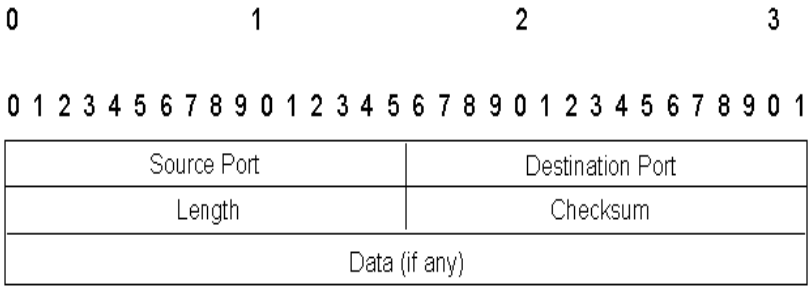
Step	Path	TCP Bit	Message
1	A -> B	SYN	My sequence number is X.
2	A <- B	ACK	Your sequence number is X.
3	A <- B	SYN	My sequence number is Y.
4	A -> B	ACK	Your sequence number is Y.

Packets 2 and 3 are normally combined into a single "SYN ACK" packet This is called a three-way handshake.

The SYN bit is set, with no ACK bit set, to show a TCP connection request. By blocking packets with the SYN bit set in a single direction, you may permit TCP connections in a single direction.

User Datagram Protocol (UDP) Level

Figure 5-4



* RFC-768 [UDP]

The UDP header fields available are the source and destination port numbers.

UDP does not permit a router to differentiate between inbound packets requesting new services and inbound packets returning data to outbound requests. This means that static filters which allow inside users access to services based on UDP also allow outside users to access the same service inside your network.

Building a Stanza - General

Some general concepts to remember when writing a stanza are listed here:

- Each stanza includes one or more numbers, addresses, or keywords, separated by slashes (/).
- Each number, address, or keyword adds an additional qualifier to the stanza.
- Each qualifier or pattern must match for the rule to be applied to the packet.
- A stanza can be used to either permit or deny a matching packet.
- Begin with the negation operator (!) to block or deny a packet.
- Any stanza that is not negated permits matching packets.
- Comments begin with a '#' character and extend through the end of the line.

Building a Stanza - Specifics

The section below explains how different features of a stanza should be written and the ways you include them in a stanza to affect the operations of the filter. The features are described in subsections which include a general explanation, an example, and a comment on the action caused by the example. The comments are shown on the same line as the example and begin with a '#' character.

We recommend that readers who are unfamiliar with filtering take the time to read this section from beginning to end. The topics and examples build on one another. Therefore, skipping through the examples, you may see references to keywords you do not recognize or miss the significance of relationships between some keywords. These topics are covered:

- Numbers and addresses
- Keywords
- Directions of packets
- Time based restrictions
- ICMP messages
- Logging and tracing

Numbers and Addresses

A number by itself, without an associated protocol name such as tcp or udp, represents an IP protocol number. IP protocol numbers have a range of 0-255 and are assigned by the Internet Assigned Number Authority (IANA). The current list of IP protocols can be found in the Assigned Numbers RFC.

Example:

```
!89  # block Open Shortest Path First (OSPF) Interior Gateway
     # Protocol (IGP)
```

IP Addresses

An IP address may be represented in hexadecimal (for example, 0xc0000201) or dotted quad (for example, 192.0.2.1) notation and represent either a host address or a network address. Network addresses are simply used to represent contiguous ranges of hosts and therefore do not necessarily correspond to actual networks. A network address uses all zeros for the host portion of the address.

Example:

```
!192.168.199.1     # block packets to/from host 192.168.199.1
```

Netmasks

You must specify a netmask if the host portion of the network address does not match the "natural" netmask. A netmask may be represented in either hexadecimal or dotted quad notation. A network address must also be present or the netmask is assumed to be an IP address.

Example:

```
10.7.123.0/255.255.255.0     # permit 10.7.123.0-10.7.123.255
```

Alternatively, you may specify the netmask after the address using a semicolon followed by the number of one bits in the network mask.

Example:

```
10.7.123.0;24     # permit 10.7.123.0-10.7.123.255
```

Keywords with Numbers

A number of keywords can describe features of the packets, including data within the packet header and the direction of travel.

IP Protocol Keywords

Keywords exist for the most commonly used IP protocols, ‘tcp’, ‘udp’, and ‘icmp’. Use only one protocol keyword per stanza. Use these keywords to prevent ambiguity when specifying port numbers or types. You must use ‘17/tcp’ or ‘6/udp’ to avoid ambiguity.

Port Numbers

‘udp’ or ‘tcp’ combined with a number represents a port number.

Example:

```
25/tcp     # permit SMTP mail
```

You may specify a range of ports using a hyphen-separated pair of numbers.

Example:

```
!0-1023/udp     # block privileged UDP ports
```

Port Numbers and Services

Many systems provide a list of well-known UDP and TCP port numbers in a services file or they supply contents of the file through a database service such as NIS or NetInfo. Filter stanzas may use the symbolic names for these port numbers.

Example:

```
smtp     # permit SMTP (25/tcp) service
```

Use keywords to identify services that are supported by more than one protocol. Symbolic port names (i.e., services) can be ambiguous. For example, 'domain' may be either 'tcp' or 'udp' port 53. When using keywords that are specific to a protocol like TCP, you must add the 'tcp' keyword to the stanza to avoid errors.

Example:

```
!domain/tcp
```

Services are not the same as applications or protocols. Note that the Telnet application is not the same as the telnet service. Specifying '!telnet' in the filter file does not prevent the 'telnet' application from talking to the 'smtp' port on the host, nor does it prevent someone from using the telnet protocol to talk to a telnet daemon (telnetd) running on a port number other than 23.

Numbered ICMP Messages

'icmp' with a single number represents an ICMP message type. 'icmp' with two numbers represents a specific ICMP type and code combination. ICMP type and code values can be found in the Assigned Numbers RFC or in the `/usr/include/netinet/ip_icmp.h` file directory on many systems. More information on numbered ICMP messages is included in the Unreach Keyword section.

Example:

```
!icmp/5     # block ICMP Redirect messages
```

Example:

```
!3/0/icmp     # block ICMP Unreachable "bad net" messages
```

Keywords with Origins and Destinations

Frequently, a host acts as a router between two end points so packets may not originate or terminate at that host. Use the 'src' keyword to specify the point of origin or source and the 'dst' keyword to specify the endpoint or destination. The source or destination can be either an IP address or a service or port.

Example:

```
route/dst   # permit packets to UDP port 520 on any host
```

The 'src' or 'dst' keyword applies to all addresses and ports in the stanza.

Thus, you cannot specify both the source port and destination address or vice versa in the same stanza using the 'src' or 'dst' keywords.

Example:

```
10.0.0.1/route/dst   # permit UDP packets to 10.0.0.1 port 520
```

Even more specific rules may be written by using the keywords 'srcport', 'dstport', 'srcaddr', 'dstaddr', 'srcmask', and 'dstmask' keywords in place of 'src' or 'dst'. The syntax of these keywords is 'keyword=value'.

Example:

```
 srcaddr=10.7.127.0/srcmask=255.255.255.0/dstaddr=192.168.5.0
            # block packets between the 10.7.127.0-10.7.127.255 and
            # 192.168.5.0-192.168.5.255 networks
```

The workstation running the daemon is your point of reference regarding the direction of a packet. Packets written by the host are outbound or sent and are specified by the 'send' keyword. Packets read by the host are inbound, or received, and are specified by the 'recv' keyword. The two rules in the following example prevent spoofed packets for an internal network 192.168.12.0:

Example:

```
!recv/src/192.168.12.0  # block receiving packets from outside which claim to
                        # be from 192.168.12.0-192.168.12.255
!send/dst/192.168.12.0  # block sending packets to outside which claim to be
                        # going to 192.168.12.0-192.168.12.255
```

Keywords Based on TCP Packet Header Bits

Only one TCP field can be specified in a rule; specifying more in the same rule is a syntax error.

Some qualifiers (keywords) may only be used in combination with other qualifiers. For example, 'syn', 'fin', 'rst', 'ack' and 'estab' are options or fields in TCP packet headers and may only be used when the qualifier 'tcp', is directly stated in the rule or implied by a TCP protocol service. If the definition of a service allows it to use TCP and UDP packets the 'tcp' qualifier must be explicitly added to the service name in the rule. After migration, a dialout PPP connection is defined by:

Example:

```
tcp/syn/recv   # block inbound TCP connection requests
smtp/syn       # permit SMTP (25/tcp) connection requests
domain/tcp/syn # permit DNS (port 53) TCP connection requests
```

'syn' Qualifying Keyword

A rule that qualifies a session with 'recv' or 'send' prevents the session from being started or logged unless it is initiated in the indicated direction. The initiator sends a SYN packet to the recipient to open a TCP data stream. This permits the filter to distinguish between outbound and inbound uses of TCP applications such as telnet or FTP. The special keyword 'syn' allows you to filter or log these connection starters. Unlike most other qualifiers, 'syn' is actually a compound qualifier that tests for just the initial packet, which has a SYN bit set but not the ACK bit.

Using Other Keywords Based on TCP Packet Header Fields

- The special keyword 'estab' identifies any TCP packet that does not have the SYN bit set. 'estab' is to 'existing' as 'syn' is to 'beginning'.
- The special keyword 'ack' allows you to filter or log the packets that have the ACK bit set.
- The special keyword 'rst' allows you to filter or log the packets that reset TCP connections.
- The special keyword 'fin' allows you to filter or log the packets that close TCP connections.

Keywords Based on IP Options

The 'ip-opt=' keyword can be used to select packets based on whether they bear various IP options, including those described in the table below:

OPTION	DESCRIPTION
rr	Record Route is used to trace the route an internet datagram takes.
ts	Time Stamp.
security	Security is used to carry Security, Compartmentation, User Group (TCC), and Handling - Restriction Codes compatible with DOD requirements.
lsrr	Loose Source Routing is used to route the internet datagram based on information supplied by the source
satid	SATNET Stream Identifier (obsolete).
ssrr	Strict Source Routing is used to route the internet datagram based on information supplied by the source.
srcrt	Either Loose Source Routing or Strict Source Routing.
any	Any IP option including the 'No Operation' option.

Example:

```
!ip-opt=srcrt    # block source routed packets
```

frag Keyword

The 'frag' keyword permits filtering of IP fragments. When IP packets are larger than the media permits, the datagram can be fragmented into smaller segments, transmitted, and reassembled at the destination. IP datagrams can be up to 65,535 bytes long, but most physical networks do not support datagrams that large. Ethernet supports datagrams 1500 bytes long. SLIP's default is 1024 bytes.

all Keyword

The special keyword ‘all’ matches any packet. It typically appears at the end of a filter to either permit or block all unspecified packets. The software automatically implicitly adds ‘!all’ at the end of a stanza list if the last stanza is not negated, and ‘all’ at the end of a stanza list if the last stanza is negated. While not strictly necessary, it is a good idea to explicitly state your preference.

Time-Based Keywords

You may add time-based restrictions to your packet filter, limiting the types of traffic passing through the packet filter during the workday, or enabling additional access after business hours or on weekends.

Two keywords define time-based rules. They are the ‘weekday’ and ‘daytime’ keywords.

The syntax for using the keyword ‘weekday’ in a qualifier is 'weekday=when', where ‘when’ is a string indicating the days the rule should be applied. The ‘when’ portion may be a list containing any of ‘Su’, ‘Mo’, ‘Tu’, ‘We’, ‘Th’, ‘Fr’ or ‘Sa’.

Example :

```
ftp/syn/send/weekday=MoTuWeThFr
         # prevents any weekday outbound ftp connection requests
```

The syntax for the keyword ‘daytime’ qualifier is ‘daytime=time’, where ‘time’ is a string that indicates the hours of a day the rule should be applied. You may indicate hours in a range (for example, ‘daytime=0800-1800’).

Example:

```
ftp/syn/send/weekday=MoTuWeThFr/daytime=0800-1800
                # prevents weekday outbound ftp connection requests
                # between 8am and 6pm.
```

Unreach Keyword and Sending ICMP Messages

In addition to permitting the packet to be passed or blocked, keywords also allow you to specify additional actions to take with any packet that matches the template. For example, you can silently drop the packet and/or send an ICMP message to notify the sender of the action.

The ‘unreach=’ keyword causes an ICMP Destination Unreachable message to be sent to the packet's source address, bearing the indicated code field. The ICMP Code may be specified numerically or mnemonically. A list of the messages appears on the following page. See the footnotes below for information on related RFCs.

Example:

```
ip-opt=srcrt/unreach=srcfail   # block source routed packets and
                               # notify sender of failure
tcp/113/unreach=1         # block RFC1413 Identification Protocol
                          # packets and send a Destination
                          # unreachable host message.
```

The currently available mnemonic codes are:

#	Name	Description
0	net	The destination network is unreachable.
1	host	The destination host is unreachable.
2	prot or protocol	The designated transport protocol is not supported.
3	port	The designated transport protocol (e.g., UDP) is unable to demultiplex the datagram but has no protocol mechanism to inform the sender.
4	needfrag	Fragmentation is needed and the Don't Fragment flag is set.
5	srcfail	Source route failed.
6	net-unknown	The destination network is unknown. This code normally should not be generated. It implies that the destination network does not exist. Code 0 (Network Unreachable) should be used in place of Code 6.
7	host-unknown	The destination host is unknown.
8	(see Note 1)	The source host is isolated. Routers should not generate Code 8; whichever of Codes 0 (Network Unreachable) and 1 (Host Unreachable) is appropriate should be used instead.

#	Name	Description
9	(see Note 2)	Communication with the destination network is administratively prohibited. This code was intended for use by end-to-end encryption devices used by U.S. military agencies. Routers should use the newly defined Code 13 (CommunicationAdministratively Prohibited) if they administratively filter packets.
10	(see Note 2)	Communication with the destination host is administratively prohibited. Same reasoning as message 9 above.
11	net-tos	Destination network unreachable for the designated type of service.
12	host-tos	Destination host unreachable for the designated type of service.
13	prohibited	Communication Administratively Prohibited.
14	precedence	Host Precedence Violation.
15	precedence-cutoff	Precedence cutoff in effect.
	rst	This is a special keyword which will not send an ICMP Destination Unreachable message but instead a TCP RST packet.

Note 1: RFC 1812 deprecates the use of messages 8 through 10 so their mnemonic codes have been removed. They can still be used by specifying them numerically.

Note 2: The uses of ICMP Destination Unreachable messages have grown. The list of message codes and their meanings is spread across a number of RFCs. ICMP Destination Unreachable messages are covered in RFC 792, RFC 1122, and RFC 1812.

Log and Trace Keywords

Use the keywords ‘log’ and ‘trace’ to log actions taken by the packet filter or dump the contents of the matching packet (in hex) to the system log.

Example 1:

```
frag/trace  # block and dump the contents of any IP fragment
            # received
```

Example 2:

```
tcp/syn/send/log  # pass and log all outbound TCP connection
                  # requests
```

There are two ways to invoke the 'log' keyword. The log filter allows you to define, in one location, all the packets you wish to log. Or you can add the log keyword to the individual rules in the 'pass' filter as shown in the above examples. Generally, it is easier to add a rule in the log filter requiring all rejected packets be logged rather than adding a 'log' qualifier to all rules that block packets.

Stanza Syntax

The syntax of a stanza is very flexible. In general, the order of the values and keywords in the stanza are not fixed, although certain combinations of keywords are not allowed. The rules that do exist are:

- Each value and keyword in a stanza must be separated by a single '/'.
- In a negation, the '!' must be the first character of the stanza.
- When a network mask accompanies a network address, the mask must follow the address.
- Since white space created with a space, tab, or newline entry separates stanzas, no white space is permitted within a stanza.
- Only one protocol (for example, tcp/udp/icmp) may be specified per stanza.
- syn, fin, ack, rst, or estab may not appear in the same stanza.

The following rules describe the same packet filtering, and would cause the same template to be invoked.

Example 1:

```
!telnet/syn/recv/192.0.2.0/255.255.255.0
```

Example 2:

```
!192.0.2.0/recv/syn/255.255.255.0/telnet
```

Writing a Stanza - A Complex UDP Example

The following section describes the writing of a rather complex packet filter involving the Domain Name System (DNS). It provides a good example of why you need to understand the applications in use when writing packet filters. It also shows the difficulty of writing static packet filters for UDP packets without permitting inbound network access in order to permit outbound service.

A brief explanation follows each rule. The explanation attempts to illustrate the security considerations which prompted the creation of the rule.

An Unsafe Domain Name System Rule

Your security policy should allow access for packets that need to cross the link to fulfill your needs, while still keeping out as many unrelated packets as possible. Look at the following example of a rule concerning domain name queries. It is one of easiest rules you could add to permit domain name queries, but it is also the most insecure.

```
1. domain/udp
```

Think of the stanza as though it were translated into this simple pseudo-code:

```
if
   protocol is UDP AND
   source or destination port is domain (53)
then
   permit the packet to pass
```

This means that a user on an outside host sending a UDP packet from port 53, could reach *any* UDP destination port on *any* host on your local network, including privileged ports used by other services. This would not be safe.

What Happens During Domain Name Queries

In the simplest case for domain queries, hosts on your network sends all 'domain' requests to your domain name server. Your server checks to see if it has the information cached. If not, it queries other domain name servers on the Internet to obtain the information.

The packet exchange looks similar to those shown below, where the entries mean the following:

- 'dns' is the IP address of the domain name server
- 'domain' is UDP port 53
- 'any' is any IP address on the inside or outside network (as appropriate)
- arrows represent the direction of travel

```
dns.domain -> any.domain   # the outbound domain request
dns.domain <- any.domain   # the inbound response to the request
```

In appearance, this is similar to an actual log file entry. The diagram explains fields in a normal log entry.

```
udp 192.168.199.11/domain -> 10.0.5.1/domain 124
 ^   ^              ^     ^  ^          ^      ^
 |   |              |     |  |          |      |
 |   |              |     |  |          |      + packet size (bytes)
 |   |              |     |  |          + foreign port
 |   |              |     |  + foreign address
 |   |              |     + direction
 |   |              + local port
 |   + local address
 + protocol
```

Developing Safer Domain Name Request Rules

An Exercise in Simplifying Rules

This section illustrates a process you might go through to develop domain name rules. Throughout the steps, the illustration includes two paths you might take. The first shows how individual rules might be developed. The second shows how the first rules can be condensed with the same results. After each group of rules, we will evaluate the progress, considering loopholes that might be created by the way the rules have been written. Throughout the examples we will use 192.168.199.11 to represent the IP address of the domain name server.

Step 1 - Handling Domain Name Requests

The first example attempts to map each packet description into a separate rule to permit these packets through the pass filter. This results in two rules similar to the following:

```
a) udp/srcaddr=192.168.199.11/srcport=domain/send/dstport=domain
   udp/dstaddr=192.168.199.11/dstport=domain/recv/srcport=domain
```

Alternatively, group (a) can be combined and simplified into a single rule similar to:

```
2)  udp/192.168.199.11/dstport=domain/srcport=domain
```

Both the outbound request and the inbound response match this template and no packets to UDP ports other than domain (53) are permitted by the rule.

Domain Name Requests from Other Domain Name Servers. In general, domain name servers on the Internet also want to query your domain name server to obtain information. The packet exchange is similar to the outbound request but reversed in order:

```
dns.domain -> any.domain  # the inbound domain request
dns.domain <- any.domain  # the outbound response to the request
```

This means that both the individual rules in example (a) and the combined rule in example (2), which were created to handle requests from outside hosts, are still functionally correct.

Step 2 - Rules for Domain Name Requests from Applications

There is a limitation associated with rules (a) and (2) concerning domain name queries. It arises because not all queries from the Internet will come from a domain name server. Some will come from applications, such as 'nslookup' or 'host' that use an unprivileged port (a port in the range of 1024-65535).

The packet exchange for such an inbound domain query resembles the following, where 'unpriv' is any port from 1024-65535.

```
dns.domain -> any.unpriv  # the inbound domain request
dns.domain <- any.unpriv  # the outbound response to the request
```

Because the query comes from an unprivileged port you need to add two additional rules to example (a) to cope with the new packets. You now need four stanzas to deal with domain queries.

```
(b) udp/dstaddr=192.168.199.11/dstport=domain/recv/srcport=1024-65535
    udp/srcaddr=192.168.199.11/srcport=domain/send/dstport=1024-65535
```

In this situation, if you combined the original rules as in example (2), it can be modified to permit the inbound request by removing the restriction on 'srcport'. However, the revised rule (2) still blocks the outbound response, causing the query to fail. To permit the outbound packet, you must add another rule following rule (2). Call it rule (3).

```
(2)  udp/192.168.199.11/dstport=domain
(3) udp/srcaddr=192.168.199.11/srcport=domain/send/dstport=1024-65535
```

Comparing the examples (a) and (b) to examples (2) and (3), you have reduced the number of rules needed from 4 to 2.

When trying to simplify, it is important you double check your assumptions. You should not use "udp/192.168.199.11/srcport=domain", although it would permit the query to succeed, because it also matches packets that should not be permitted. That rule is similar in function to this pseudo code that follows it.

```
(2) udp/192.168.199.11/srcport=domain

if
     protocol is UDP AND
     source or destination IP address is 192.168.199.11 AND
     source port is domain (53)
then
     permit the packet to pass
```

This means an outside host sending a UDP packet from port 53 to 192.168.199.11 can access ANY destination port, including privileged ports used by other services. This would not be "safe".

Similarly, if an internal user on the domain name server uses an application such as 'nslookup' to send a query directly to another domain name server, rather than sending the query to the local domain name server, the packet traces and rules needed to deal with them will change.

```
dns.unpriv -> any.domain  # the outbound domain request
dns.unpriv <- any.domain   # the inbound response to the request
```

Step 3 - Matching Packets as Closely as Possible

If you want your templates to match the packets as closely as possible, you must add two further rules (c) to (a) and (b). You now have 6 rules if you have not attempted to condense the rules.

```
(c) udp/srcaddr=192.168.199.11/srcport=1024-65535/send/dstport=domain
    udp/dstaddr=192.168.199.11/dstport=1024-65535/recv/srcport=domain
```

If you have simplified rules (a) and (b) and have used rules (2) and (3), the additional packets also cause rule (3) to block some of the necessary traffic. This is because the inbound packet, while originating from the

domain port, does not have a source address of the domain name server. After making the change to accommodate the differences, by removing the restriction on direction ('send') and relaxing the restriction on address, you have replaced rule (3) with the edited rule resembling (4).

```
(2) udp/192.168.199.11/dstport=domain
```

```
(4) udp/192.168.199.11/srcport=domain/dstport=1024-65535
```

Step 4 - Minimizing External Control of Data Passing through the Packet Filter

Up to this point the rules have always been based on the trust of data under local control. With static filtering, the second rule now permits data that is under external control through the packet filter.

```
if
     protocol is UDP AND
     source or destination IP address is 192.168.199.11 AND
     source port is domain (53) AND
     destination port is in the range 1024-65535
then
     permit the packet to pass
```

This means that an external host can send UDP packets from port 53 to any unprivileged port on the domain name server without requiring an internal initiator. Unfortunately, there are a number of assigned ports, such as the normal default NFS port (2049/udp), which are allocated out of the unprivileged port range and can present security problems. You can minimize the risk by reducing services on your domain name server and by adding additional rules to block access to those services before the second rule. If you explicitly block access to the port first, the packet will not reach the rule that gives it permission to pass because the filter stops as soon as a match is found

Finally, an internal user on a local host other than the domain name server may use the 'server' command of 'nslookup' to change the default server. In this case, the packet is not sent to the local domain name server but directly to the external domain name server.

```
any.unpriv -> any.domain  # the outbound domain request
any.unpriv <- any.domain  # the inbound response to the request
```

Once again, permitting the traffic to pass requires adding two additional rules, labeled (d) in the example, to group (a) (b) and (c). Now there are a total of 8 rules.

```
(d) udp/srcaddr=192.168.199.0/srcport=1024-65535/send/dstport=domain
   udp/dstaddr=192.168.199.0/dstport=1024-65535/recv/srcport=domain
```

The condensed version of the rules also requires that rule (4) be modified to permit proper operation. The modification is required because the outbound packet has a destination port 'domain' , but it is not to or from the IP address of the domain name server (192.168.199.11). After removing the IP address restriction, the final set of simplified rules is:

```
(2) udp/dstport=domain
(5) udp/srcport=domain/dstport=1024-65535
```

Conclusion

Domain name service offers a strong example of the tradeoff between functionality and security. It also illustrates the complexity of maintaining security. The level of service you decide to offer should strongly affect the rules you use. Your security policy should dictate the level of service you offer. You should not, in general, let the desired functionality dictate your security policy.

Writing a Stanza - TCP Examples

To open a TCP data stream, the initiator sends a packet to the intended recipient. The SYN bit (with no ACK bit) is set in the TCP header to show a TCP connection request. The special keyword 'syn' matches packets that have the SYN bit set, but no ACK bit set. This allows you to filter or log packets that start connections.

The TCP protocol requires more than a single SYN packet for a TCP connection to work. This means you cannot enable TCP connections with a single 'syn' rule.

Two Approaches to Filtering TCP connections

There are two approaches to filtering TCP connections. The first approach is to block 'syn' packets that you do not want to establish a service, while permitting all other packets for the service.

Example:

```
!telnet/syn/recv   # block inbound telnet connection requests
telnet             # permit all other telnet packets
```

The second approach is to permit the 'syn' packets you do want to establish a connection, followed by permitting any other non-SYN packets. The opposite of using the 'syn' keyword is the 'estab' keyword. 'estab' describes any TCP packet that does not have the SYN bit set or that has both the SYN and ACK bits set in the TCP header.

Example:

```
telnet/syn/send    # permit outbound telnet connection requests
telnet/estab       # permit packets to established connections
```

Identifying Rulesets with Hostnames and Addresses

The first line of a ruleset must contain a hostname, IP address, or 'default' that identifies the interface you wish to use the ruleset. For pppd, use the peer, or remote, IP address. This hostname or IP address must not be indented because the name/address will be assumed to be part of a rule and may or may not cause a syntax error.

We recommend that you use an IP address, but if you use a hostname, it is important that the system can resolve the hostname locally. If the link must be up to resolve the name, the hostname matching fails and the interface is forced to use the default ruleset. This changes the meaning of your filter file and causes long delays because the connection times out while waiting for name resolution.

A Note on Ruleset Formatting

Each additional line of a ruleset (continuation lines) must be indented, using one or more white space characters (space or tab). If a line is not indented, the first word on the line is assumed to be a hostname for a new ruleset.

Any information that follows a '#' character on a line is assumed to be a comment and is ignored.

Ordering Stanzas Effectively

When a ruleset has been selected, each stanza is applied in order, top to bottom, and left to right. The filter returns when it finds a matching stanza. It is therefore important that you order stanzas in the Filter file in most specific to least specific (most general) order. If the order is reversed, the most general rule matches first and the filter never reaches the more specific rules.

Example of correct order:

```
domain/192.0.2.1          # permit DNS to/from 192.0.2.1
!domain                   # prevent all (other) DNS
```

Example of wrong order:

```
!domain                    # prevent all DNS
domain/192.0.2.1           # this rule is never reached
```

Isolating an 'Incorrect' Stanza

When pppd finds an error in the Filter file, the line number that caused the failure is reported. When a packet is rejected due to a filter and the 'log rejected' keyword is used, the line number that caused the rejection can be recorded in the log file. Therefore, when tracking down problems, you can quickly isolate the correct stanza if only one stanza is used per line.

Working with Default Rulesets

The default ruleset is:

```
default bringup all pass all keepup all log !all
```

This is probably an unacceptable default if you are trying to filter packets. Your default should be the same as your most restrictive ruleset because it keeps your site secure if connection-specific filtering fails due to a misconfigured IP address or hostname.

The following sections deal with two approaches to writing default rulesets. The difference between the two approaches is apparent from their names, the "closed policy" and the "open policy."

Open Policy Default Rulesets

If you are willing to accept most packets from a site from which you have not previously accepted traffic, a reasonable default filter might be:

```
default
        bringup all
        pass !exec !tftp all
        keepup  all
        log rejected !all
```

Notice the use of 'all' rather than '!all' at the end of each filter. This default ruleset only blocks 'tftp' packets and 'rexec' packets, two protocols that normally should never cross organizational boundaries.

A Note on Using the 'log rejected' Filter

In the previous example, and in the following Closed Policy examples, we use the 'log rejected' filter. This is a good default log filter to use. When testing, it permits you to see that your filter is working as expected and keeps track of outside attempts to connect through to blocked services.

Closed Policy Default Rulesets

The closed policy default rulesets in the examples that follow illustrate the way services can be safely and incrementally allowed for a remote site. This approach begins from the premise that no packets should be allowed to pass.

Block All Packets

If your security policy changes and you must block packets from a formerly acceptable site, you might change the default filter to the following:

```
default bringup !all pass !all keepup !all log rejected !all
```

Block All Packets Except Electronic Mail

Most sites are willing to permit electronic mail through the workstation or system. For this, and the following examples, it is important that you have the smtp service defined in your services file. To help prevent any problems due to this assumption, you could use the explicit port number, protocol, and IP address to test. Then you might write the default this way:

```
default bringup !all pass 25/tcp !all keepup !all log rejected !all
```

The filter file is easier to read and comprehend if you use the service name. Still, specifying the explicit port number and protocol can also prevent someone from changing the meaning of your rulesets by subverting NIS. The benefit may be negligible, though, if the information is only checked against the local services file. Intruders able to modify the services file could also modify the filter file.

Limiting Electronic Mail to a Gateway

If you wish to limit electronic mail access to a gateway machine, you need to add the qualification to the smtp stanza. The next two examples use the fictitious name/address bignfast/192.0.2.1.

```
default bringup !all pass smtp/bignfast !all keepup !all log rejected !all
```

Unresolvable Hostnames and Changing IP Addresses

In the last example we used the hostname 'bignfast'. But consider the inherent problem of using a hostname instead of the IP address. If the host address cannot be resolved, you may reach a state of "deadlock" because the name must be resolved for the service to begin, but the service must begin for the name to be resolved. The example below may be more reliable.

```
default bringup !all pass smtp/192.0.2.1 !all keepup !all log rejected !all
```

However, nothing is perfectly reliable. Being explicit can cause other problems if you change the address of the server. The reliability of using an IP address instead of a hostname, or vice versa, may only be decided on a site-by-site basis. Still, we strongly favor using IP addresses over using hostnames. One benefit is that specifying the explicit IP address prevents people from changing the meaning of your rulesets by DNS spoofing.

Conclusion

Build up the list of what you would let an unknown site do a little at a time as you discover services you wish to allow. The important thing is to remember to place '!all' at the end of each filter. You and any future manager can quickly see you are blocking all but specific packets and you reduce the chance that someone will accidentally change the meaning of the implicit ending stanza to 'all'.

A Note - Blocking Loose Source and Strict Source Routing Options

Using the IP Source Routing options, it is possible for people to send packets to you that look like they are coming from a host on your network. Prevent this sort of attack by blocking packets that have the Loose Source Routing or Strict Source Routing IP options set. If you want to be restrictive, add the line below to all your rulesets, including your default ruleset in the default pass filter:

```
!ip-opt=srcrt
```

Closed Policy Filter Example

The following is an example static filter configuration, appropriate for a system using pppd to create a PPP/SLIP link between the system 192.168.199.1 and a peer, 10.0.0.1, that is acting as the gateway to the Internet. The complete filter, minus the comments, follows this section.

The filter design reflects a fail-safe, or closed, policy.

Default

```
default
    pass !all          # block all other packets
    log  rejected      # packets rejected by packet filter
```

First, we define a default ruleset that is very restrictive. This is a failsafe ruleset that will not pass any packets through the filter, but will notify you of all traffic you are missing.

```
10.0.0.1
```

This ruleset will be applied to any packet crossing the link connecting this host to the peer (10.0.0.1).

```
    bringup
        !3/icmp          # ICMP unreachable messages
        !5/icmp          # ICMP redirect messages
        !11/icmp         # ICMP time exceeded messages
        !who             # WHO service (513/udp)
        !route           # routed/gated RIP service (520/udp)
        !ntp             # Network Time service (123/udp)
         all             # all other packets
```

If the link is configured for 'dial on demand' connections, the 'bringup' filter describes those packets that causes a call to be placed and a connection to be initiated. The 'bringup' filter should be used to prevent the connection from being brought up inappropriately. It is a good idea to block packets that are responses to "bad" inbound packets, such as ICMP Destination unreachable messages, because they aren't "interesting" enough to dial the modem. You should also block services, such as the WHO service, that send packets at a regular intervals and would therefore never permit the link to stay down long. Any other sort of traffic initiates a dial connection.

```
pass
   !recv/ip-opt=srcrt/unreach=srcfail  # Block SRCRT attacks
```

Do not allow any incoming packets with the Source Route option set in the IP header. Respond with an ICMP Destination Unreachable message with the Source Route Failed code value.

```
!192.168.199.0/recv/src/unreach=net # Block IP spoofing attacks
!192.168.199.0/send/dst/unreach=net # Block IP spoofing attacks
```

Block any incoming packets that claim to be from your net, and block any outgoing packets that claim to be destined for your net. Respond with an ICMP Destination Unreachable message with the Bad Net code value.

```
!127.0.0.0;8                # Block IP spoofing attacks
```

Silently block all packets that claim to be either to or from the loopback network.

```
dstport=nntp/dstaddr=192.168.199.10/srcaddr=10.0.5.6
dstport=nntp/srcaddr=192.168.199.10/dstaddr=10.0.5.6
dstport=nntp/dstaddr=192.168.199.10/srcaddr=172.31.12.13
dstport=nntp/srcaddr=192.168.199.10/dstaddr=172.31.12.13
!nntp/unreach=rst
```

Allow Network News (Usenet) exchanges with only your known news neighbors (10.0.5.6 and 172.31.12.13) and your news server 192.168.199.10). Block any other NNTP traffic, and respond with a TCP RST message.

```
domain/tcp/192.168.199.11/dst/syn/recv  # (53/tcp)
!domain/tcp/syn/recv
domain/tcp/192.168.199.11
```

Allow outside hosts to obtain Domain Name Service zone transfers only if your end of the stream is really being handled by your domain name server. In this example, you first permit inbound requests to the domain name server, then block all other inbound requests, and finally allow any TCP packets to pass over the link if they are to or from the host 192.168.199.11 and to or from the domain port to pass over the link. The sender will not be notified that the packets are being dropped.

```
dstport=domain/udp/192.168.199.11 # permit domain queries (53/udp)
!domain                           # block domain (53/tcp, 53/udp)
```

Allow Domain Name Service (DNS) queries to and from the DNS server. Block all other domain requests. This second rule is not strictly necessary, since the final rule is '!all', however adding this rule makes it failsafe.

```
smtp/192.168.199.14/dst/syn/recv        # (25/tcp)
!smtp/syn/recv
smtp
```

Allow incoming electronic mail connection requests to reach your SMTP server, allow no other incoming SMTP connection requests, and allow yourself unlimited outbound SMTP access.

```
www/syn/recv/192.168.199.13/dst    # (80/tcp)
!www/syn/recv/unreach=host         #
www                                 #
```

Allow incoming World Wide Web connection requests to reach your WWW server. Allow no other incoming WWW connection requests. And allow yourself unlimited outbound WWW access.

```
!dstport=ident/recv/unreach=rst # block IDENT service (113/tcp)
```

You do not use the RFC 1413 identification services, so you might as well bounce the queries at the gateway instead of having `inetd` refuse the connection. Respond with a TCP RST message. This does not improve the security of your packet filter, since the packets would be blocked by the final '!all', but it does reduce the delay in services that make use of 'ident'.

```
!telnet/syn/recv/unreach=prohibited  # block inbound TELNET
                                     # requests
telnet                               # permit TELNET messages
```

Allow outbound telnet connections from your network to anywhere else.

```
!finger/syn/recv/unreach=prohibited    # block inbound FINGER
                                       # requests
finger                                 # permit FINGER messages
```

Block incoming finger requests until you install a safe finger daemon.

```
ftp/syn/recv/dst/192.168.199.12  # permit inbound FTP for anon FTP
!ftp/syn/recv/unreach=host       # block inbound FTP
                                 # requests
```

Allow incoming FTP (file transfer) traffic that uses your Anonymous FTP server system, but block any other incoming FTP requests. Respond with an ICMP Destination Unreachable message with the Bad Host code value.

```
ftp                    # permit FTP messages
srcport=ftp-data/dstport=1024-65536/syn
!ftp-data/syn          # block other FTP-DATA connections
ftp-data               # permit FTP-DATA messages
```

After blocking the traffic specified above, allow both FTP command streams and FTP data streams to cross the link, both inbound and outbound.

```
dstport=33410-33515/udp/send # permit outbound traceroute operation
```

The traceroute tool probes high-numbered UDP ports and is so useful that you should let it through.

```
        !5/icmp                         # block ICMP_REDIRECT
        8/icmp/192.168.199.1            # permit ping of gateway
        8/icmp/192.168.199.10           # permit ping of NNTP server
        8/icmp/192.168.199.11           # permit ping of DNS  server
        8/icmp/192.168.199.12           # permit ping of FTP  server
        8/icmp/192.168.199.13           # permit ping of WWW  server
        8/icmp/192.168.199.14           # permit ping of SMTP server
        !8/icmp/recv                    # block inbound ping address
                                        # scanning
        icmp                            # permit ICMP messages
```

Block ICMP redirect messages since the routing on an internal node should not be changed by an external site. Permit ICMP echo request packets, sent by 'ping', to reach all hosts providing external services. Block all other inbound ping packets to prevent IP address probes. Finally, allow other ICMP messages to pass freely.

```
        !all                             # block all other packets
```

Silently block all traffic not explicitly permitted to pass. Pass through the firewall only those packets explicitly permitted to pass.

```
    keepup
        !send                       # outbound traffic
        !3/icmp                     # ICMP unreachable messages
        !5/icmp                     # ICMP redirect messages
        !11/icmp                    # ICMP time exceeded messages
        !who                        # WHO protocol
        !route                      # routed/gated RIP protocol
        !ntp                        # Network Time Protocol
        all                         # permit all other packets
```

The link is considered active (non-idle) if any packet passes that is not specified in the keepup filter as being blocked. Since there are certain link failure modes that allow your system to continue sending even though the peer is unresponsive, no outbound packets are permitted to reset the idle timer.

```
log
    !8/icmp                         # ICMP ECHO packets
    rejected                        # packets rejected by packet
                                    # filter
    tcp/syn                         # all TCP connection requests
    !all                            # block all other packets
```

Log any packet blocked by the 'pass' filter above, except ICMP Echo messages. Also log all TCP connection requests.

Complete Filter Example

```
default
   pass !all  # block all other packets
   log  rejected # packets rejected by packet filter
10.0.0.1
   bringup
      !3/icmp    # ICMP unreachable messages
      !5/icmp    # ICMP redirect messages
      !11/icmp   # ICMP time exceeded messages
      !who       # WHO service (513/udp)
      !route     # routed/gated RIP service (520/udp)
      !ntp       # Network Time service (123/udp)
      all        # all other packets
  pass
      !recv/ip-opt=srcrt/unreach=srcfail     # block SRCRT attacks
      !192.168.199.0/recv/src/unreach=net    # block IP spoofing attacks
        !192.168.199.0/send/dst/unreach=net  # block IP spoofing attacks
        !127.0.0.0;8                         # block IP spoofing attacks
        dstport=nntp/dstaddr=192.168.199.10/srcaddr=10.0.5.6
        dstport=nntp/srcaddr=192.168.199.10/dstaddr=10.0.5.6
        dstport=nntp/dstaddr=192.168.199.10/srcaddr=172.31.12.13
        dstport=nntp/srcaddr=192.168.199.10/dstaddr=172.31.12.13
        !nntp/unreach=rst
        domain/tcp/192.168.199.11/dst/syn/recv  # (53/tcp)
        !domain/tcp/syn/recv
        domain/tcp/192.168.199.11
        dstport=domain/udp/192.168.199.11   # permit domain queries (53/udp)
        !domain                             # block domain (53/tcp, 53/udp)
        smtp/192.168.199.14/dst/syn/recv    # (25/tcp)
        !smtp/syn/recv
        smtp
        www/syn/recv/192.168.199.13/dst     # (80/tcp)
        !www/syn/recv/unreach=host
        www
        !dstport=ident/recv/unreach=rst     # block IDENT service  (113/tcp)
        !telnet/syn/recv/unreach=prohibited # block inbound TELNET requests
        telnet                              # permit TELNET messages
        !finger/syn/recv/unreach=prohibited # block inbound FINGER requests
        finger                              # permit FINGER messages
        ftp/syn/recv/dst/192.168.199.12     # permit inbound FTP for anon FTP
        !ftp/syn/recv/unreach=host          # block inbound FTP requests
        ftp                                 # permit FTP messages
        srcport=ftp-data/dstport=1024-65536/syn
        !ftp-data/syn                       # block other FTP-DATA connections
        ftp-data                            # permit FTP-DATA messages
        dstport=33410-33515/udp/send        # permit outbound
                                            # traceroute operation
        !5/icmp                             # block ICMP_REDIRECT
        8/icmp/192.168.199.1                # permit ping of gateway
        8/icmp/192.168.199.10               # permit ping of NNTP server
```

Closed Policy Filter Example

```
    8/icmp/192.168.199.11   # permit ping of DNS server
    8/icmp/192.168.199.12   # permit ping of FTP server
    8/icmp/192.168.199.13   # permit ping of WWW server
    8/icmp/192.168.199.14   # permit ping of SMTP server
    !8/icmp/recv            # block inbound ping address scanning
    icmp                    # permit ICMP messages
    !all                    # block all other packets
keepup
    !send                   # outbound traffic
    !3/icmp                 # ICMP unreachable messages
    !5/icmp                 # ICMP redirect messages
    !11/icmp                # ICMP time exceeded messages
    !who                    # WHO protocol
    !route                  # routed/gated RIP protocol
    !ntp                    # Network Time Protocol
    all                     # permit all other packets
log
    !8/icmp                 # ICMP ECHO packets
    rejected                # packets rejected by packet filter
    tcp/syn                 # all TCP connection requests
    !all                     # block all other packets
```

Open Policy Filter Example

This example of a filter is the product of an open policy. It was developed for the same system configuration as was used in the previous example demonstrating a filter developed for a closed policy. The system uses pppd to create a PPP/SLIP link between the system, 192.168.201.1, and a peer, 10.0.0.1, that is acting as the gateway to the Internet.

`default`

Since this ruleset is declared as the default ruleset and no ruleset has been defined for 10.0.0.1, it is applied to any packet crossing the link connecting this host to any peer, including the Internet gateway (10.0.0.1).

```
bringup
    !3/icmp                  # ICMP unreachable messages
    !5/icmp                  # ICMP redirect messages
    !11/icmp                 # ICMP time exceeded messages
    !who                     # WHO service (513/udp)
    !route                   # routed/gated RIP service
                             # (520/udp)
    !ntp                     # Network Time service (123/udp)
    all                      # all other packets
```

If the link is configured for 'dial on demand' connections, the 'bringup' filter describes those packets that causes a call to be placed and a connection to be initiated. The 'bringup' filter should be used to prevent the connection from being brought up inappropriately. It is a good idea to block packets that are responses to "bad" inbound packets, such as ICMP Destination unreachable messages, that are not "interesting" enough to dial the modem. You should also block services, such as the WHO service, that send packets at a regular intervals and would therefore never permit the link to stay down long. Any other sort of traffic initiates a dial connection.

```
pass
   !recv/ip-opt=srcrt/unreach=srcfail     # block SRCRT attacks
```

Do not allow any incoming packets with the Source Route option set in the IP header. Respond with an ICMP Destination Unreachable message that has the Source Route Failed code value.

```
    !192.168.199.0/recv/src/unreach=net    # block IP spoofing
                                           # attacks
    !192.168.199.0/send/dst/unreach=net    # block IP spoofing
                                           # attacks
```

Block any incoming packets that claim to be from your net, and block any outgoing packets that claim to be destined for your net. Respond with an ICMP Destination Unreachable message that has the Bad Net code value.

```
!127.0.0.0;8                        # block IP spoofing attacks
```

Silently block all packets that claim to be either to or from the loopback network.

```
!dstport=ident/recv/unreach=rst  # block IDENT service (113/tcp)
```

You do not use the RFC 1413 identification services, so you might as well bounce the queries at the gateway instead of having `inetd` refuse the connection. Respond with a TCP RST message. This does not improve the security of your packet filter, since the packets would be blocked by the final '!all', but it does reduce the delay in services that make use of 'ident'.

```
!chargen/unreach=prohibited    # block chargen service
                               # (19/tcp,19/udp)
!discard/unreach=prohibited    # block discard service
                               # (9/tcp,9/udp)
!echo/unreach=prohibited       # block echo service
                               # (7/tcp,7/udp)
```

Block access to your "character generator" port, and two others, because they are only meant as testing facilities and could be used by someone outside to swamp your network's bandwidth with unwanted, but otherwise harmless, traffic.

```
        !5/icmp                       # block ICMP_REDIRECT
```

Block ICMP redirect messages since the routing on an internal node should not be changed by an external site.

```
!sunrpc     # block portmap (sunrpc 111/tcp,111/udp)
!exec       # block rexecd (512/tcp)
!login      # block rlogind (513/tcp)
!shell      # block rshd (514/tcp)
!syslog     # block syslogd (514/udp)
!printer    # block lpd (515/tcp)
!2049/udp   # block nfsd (2049/udp)
```

Block access to a number of services that depend upon IP addresses for authentication, or that have no authentication built-in.

```
!tftp                 # block tftp (69/udp)
```

Block access to the tftp port because it is sometimes buggy and thus permit access to files that should not be distributed, such as password files. It is normally only used locally and should not be receiving requests from outside the local area network.

```
        all                     # permit all other packets
```

Permit all traffic except that which you have explicitly specified as blocked through the firewall.

```
keepup
    !send               # outbound traffic
    !3/icmp             # ICMP unreachable messages
    !5/icmp             # ICMP redirect messages
    !11/icmp            # ICMP time exceeded messages
    !who                # WHO protocol
    !route              # routed/gated RIP protocol
    !ntp                # Network Time Protocol
    all                 # all other packets
```

The link is considered active (non-idle) if any packet passes that is not specified as blocked in the keepup filter. Since there are certain link failure modes that allows your system to continue sending even though the peer is unresponsive, no outbound traffic counts against the idle timer.

```
log
    rejected          # packets rejected by packet filter
    !all              # block all other packets
```

Log any packet blocked by the 'pass' filter above.

Common Mistakes

A number of errors are common enough to be specifically pointed out.

Incorrect Ruleset Indentation

A common mistake is to accidentally indent the ruleset identifier (for example, hostname, IP address, or 'default'). This causes the software to assume that it is a part of the previous ruleset rather than defining the start of a new ruleset.

Yet another common mistake is not indenting all the parts of a ruleset after the initial ruleset identifier. This causes the software to assume that the stanza is the start of a new ruleset. This causes both failures and/or delays as the software tries to resolve the stanza into an IP address.

Incorrect Use of 'tcp' and 'udp'

A different error is to fail to specify 'tcp' or 'udp' when a service can use either service but the keywords are applicable to only one. An example of this is the domain name service (DNS), which uses UDP for normal

queries, but TCP for zone transfers. If you try to block inbound requests for a zone transfer you must remember to add the 'tcp' qualifier to the service name 'domain' to prevent a syntax error.

Attempting to Send Hostnames Requiring Resolution over Down Network Links

You can easily (but mistakenly) use a hostname that needs to be resolved because it is is not defined or that requires DNS, over a network link that is down. This causes failures and/or delays. Therefore, we would strongly recommend the use of only IP addresses in filter files.

Failing to Allow Passage of 'ftp' Data Packets over the 'ftp-data' Port

Less commonly, some people who do not fully understand the protocol only permit 'ftp' packets to pass through their filter. The FTP protocol actually uses two channels; the first channel is used for commands and the second port is for used for data. The second channel uses a separate port, commonly 'ftp-data', but that is actually determined during the FTP negotiations. The second channel is normally a reverse channel used to transfer data back to the client.

Blocking Packets Required for Network Access

Finally, make sure you permit passage of all packets required for your network access. Some Internet Service Providers (ISPs) require the use of a routing protocol and will mark the link inactive if they do not receive routing packets. This may require a rule in the pass clause of your ruleset to permit the route packets to traverse the link (for example, 'route').

Complete Filter Example

```
default
    bringup
        !3/icmp                                 # ICMP unreachable messages
        !5/icmp                                 # ICMP redirect messages
        !11/icmp                                # ICMP time exceeded messages
        !who                                    # WHO service (513/udp)
        !route                                  # routed/gated RIP service (520/udp)
        !ntp                                    # Network Time service (123/udp)
        all                                     # all other packets pass
        !recv/ip-opt=srcrt/unreach=srcfail  # block SRCRT attacks
        !192.168.199.0/recv/src/unreach=net # block IP spoofing attacks
        !192.168.199.0/send/dst/unreach=net # block IP spoofing attacks
```

```
    !127.0.0.0;8                        # block IP spoofing attacks
    !dstport=ident/recv/unreach=rst     # block IDENT service (113/tcp)
    !chargen/unreach=prohibited         # block chargen service
                                        # (19/tcp,19/udp)
    !discard/unreach=prohibited         # block discard service
                                        # (9/tcp,9/udp)
    !echo/unreach=prohibited            # block echo service
                                        # (7/tcp,7/udp)
    !5/icmp                             # block ICMP_REDIRECT
    !sunrpc                             # block portmap (sunrpc
                                        # 111/tcp,111/udp)
    !exec                               # block rexecd (512/tcp)
    !login                              # block rlogind (513/tcp)
    !shell                              # block rshd (514/tcp)
    !syslog                              # block syslogd (514/udp)

    !printer                            # block lpd (515/tcp)
    !2049/udp                           # block nfsd (2049/udp )
    !tftp                               # block tftp (69/udp)
    all                                 # permit all other packets
keepup
    !send                               # outbound traffic
    !3/icmp                             # ICMP unreachable messages
    !5/icmp                     # ICMP redirect messages
    !11/icmp                    # ICMP time exceeded messages
    !who                        # WHO protocol
    !route                      # routed/gated RIP protocol

   !ntp                                 # Network Time Protoco
   all                                  # all other packets
   log
   rejected                             # packets rejected by
                                        # packet filter
   !all                                  # block all other packets
```

Time-To-Call Restrictions

The second field on each line in the `Systems` file specifies the times `pppd` is allowed to attempt to establish connections. Two benefits of time restrictions are:

- Assurance that there will be personnel on each end of the link to monitor connection attempt.
- Assurance that connections take place when the most favorable telephone calling rates apply.

See `ppp.Systems(4)` for details. The when field is very flexible. It can be configured by day and by hour, with many different combinations allowed for the same connection. For example, the following entry allows connections on any day between 1 AM and 6AM or any time on Saturday and Sunday:

```
lark Any0100-0600|Sa|Su ACU 38400 5551212 in:--in: Probin word: mypasswd
```

Dial-Back

PPP supports the ability to maintain a connection when calling a modem that has a dial-back security feature. The `Systems` file chat script option \M allows this by disabling delivery of SIGHUP to `pppd`. This signal usually results from loss of Carrier Detect and tells `pppd` to abruptly disconnect from the active session.

Dial-Back Process

Typically, an answering modem with dial-back capability responds to a call by taking the following steps:

1. Challenges incoming callers with a prompt string.
2. Accepts the input identifying the caller.
3. Hangs up the call.
4. Calls a number associated with the caller's identification.
5. Re-establishes a carrier.

The calling modem might then demand the same type of identification before allowing remote data to flow through its serial interface to the local system.

Blocking SIGHUP with Chat Script \M Option

The calling system's `pppd` must be prepared for the temporary lack of a Carrier Detect signal from its modem during the dial-back from the remote modem. To avoid receiving a SIGHUP, `pppd` instructs the UNIX system's serial drivers not to deliver the signal. In other words it says, "Temporarily treat the serial interface as if it were connected to a local device like a terminal or printer, instead of a modem." `pppd` does this by specifying \M in the 'send' phase of a `Systems` chat script. See `ppp.Systems(4)` for details on \M and \m chat script options.

Reversing Instructions with \m Option

After the disconnection period, through the 'send' phase option \m, pppd tells the system's serial drivers to reverse the first instruction and respect the modem's full variety of control. For example, to dial into a system protected by a dial-back modem, the Systems chat script might be written like this:

```
#
# This connects to a system protected by a Telebit T3000 callback
# modem
# with S46=2.
#
    server   Any   ACU   38400   19071234567   TIMEOUT   60   \
    ENTER\sPASSWORD:   my_modem_password\M \
    ENTER\SPASSWORD:   my_modem_password\m \
    login:   my_login_name password:   my_login_password
```

Link Peer Authentication

PPP implements both the Password Authentication Protocol (PAP) and the Challenge Handshake Authentication Protocol (CHAP). If pppd is invoked with any of the authentication options, it demands that the peer (either calling or called) authenticate itself. The ppp.Auth(4) file contains pairs of either names and secrets for CHAP negotiation, or usernames and passwords for PAP negotiation. If a peer provides a name or username, its secret or password must match that found in the Auth file or the authentication phase fails and the connection is terminated. Each name/secret pair in the Auth file may be followed by address patterns restricting the peer's negotiated IP address. If an address restriction is specified for a particular name and the peer's negotiated IP address does not match the restriction address patterns, pppd terminates the connection.

The rechap interval option instructs pppd to periodically (every interval seconds) challenge the peer to authenticate itself. If the peer fails the new challenge, the link is terminated.

Replacing getty with pppd

Incoming calls most often invoke the `getty` program because it enables login on a serial port. However, in some cases, `pppd` can be invoked in place of `getty`. Invoking `pppd` offers additional security because people find the beginning of LCP option negotiations much more difficult to circumvent than a simple 'login:' prompt. When the modem answers an incoming call and raises the Carrier Detect signal, the caller sees a burst of what looks like line noise.

To replace `getty` with `pppd`, first use SAM to set up a `getty` on the serial port. When setting up the serial port through SAM, be sure to choose "Receive Incoming Calls (start getty process)". This adds a `getty` (or `uugetty`) line to the `/etc/inittab` file. As an example, suppose the line added to `/etc/inittab` by SAM were:

```
a0:3:respawn:/usr/lbin/uucp/uugetty -r -t 60 -h ttyd0p1 19200
```

You can replace the `uugetty` with the `pppd` process and invoke it with any arguments you like. For example:

```
a0:3:respawn:/usr/bin/pppd localhost: idle 120 requireauth ttyd0p1 19200
```

It should be noted that the device (ttyd0p1) and speed (19200) should be specified on the `pppd` command line as `pppd` needs to know what device to open and at what speed.

Note that Challenge Handshake Authentication Protocol (CHAP) is still strongly recommended for security.

6 Troubleshooting pppd

This troubleshooting chapter describes solutions to common problems.

Scenario: pppd dials, the modems connect, the modem data lights flash briefly, and then the log file shows 'Hangup' or 'SIGHUP.'

Solution: Increase pppd's debug level to 2 (chat script processing) to see if any error messages are printed by the peer at startup. 'Login incorrect' indicates that the login and/or password in the calling machine's Systems file don't match those in the answering machine's passwd file. 'etc/ppp/Login:Permission denied' indicates a file protection problem with the login shell script. '/usr/etc/pppd: Permission denied' indicates that the PPP user account cannot execute the daemon, probably because the user account is not in the 'ppp' group that owns the daemon, while the daemon is mode 4750 for security reasons.

Scenario: pppd dials out, the modems connect, the log file says 'Call succeeded,' but ps shows pppd's state to be 'connecting' until the modems disconnect about a minute later.

Solution: Check the log file for messages indicating negotiation problems. Increase pppd's debugging level to 2 (input framing errors). Look in the pppd.log file for warnings indicating damaged messages, such as 'Bad FCS received,' 'Frame too long,' or 'Missed ALLSTATIONS.'

Increase pppd's debugging level to 9 (show all characters read or written) to see if the peer is sending anything at all. PPP messages should initially look like 7E FF 7D 23 ... 7E. Look for corrupted characters (for example, 7F instead of FF) or other parity bit problems.

Scenario: PPP connects, and the user can ping the other end, but cannot use telnet, rlogin, or ftp.

Solution: Check the log file for any messages indicating negotiation problems. Increase pppd's debugging level to 2 (input framing errors). Look in the pppd.log file for warnings indicating damaged messages, such as 'Bad FCS received,' 'Frame too long,' or 'Missed ALLSTATIONS,' 'Bad protocol,' or 'Short frame received.'

Increase pppd's debugging level to 6 (IP message summary). When telnet is started, make sure a TCP packet is sent. A TCP packet should be received in response.

Restart pppd with the novjcomp option. If this fixes the problem, then the peer may be using a buggy IPCP option negotiation algorithm. Try running pppd with the rfc1172-vj or the rfc1172-typo-vj option. If these options do not work, go back to novjcomp and complain to the vendor of the peer's implementation of PPP.

Scenario: PPP connects and everything works, but it is slow and erratic.

Solution: Check if your Telebit modem is using the PEP protocol. PPP over PEP is normally show and erratic.

Increase pppd's debugging level to 2 (input framing errors). Make sure that received frames are arriving undamaged and that you are not getting an excessive number of FCS errors.

Make sure that the telephone lines are relatively clean. Noisy phone lines can cause FCS or other transmission errors, or can cause V.42 or MNP error correcting protocols to retransmit excessively, or can cause the modems to retrain frequently, slowing the connection.

Make sure the serial port is not running faster than your computer can handle.

Make sure that flow control is set up properly on the modem and on the serial port to which it is connected. If you are using hardware flow control, make sure the cable connects RTS and CTS (RS-232 pins 4 and 5).

Scenario: When the user uses telnet, ping, or some other program to attempt a connection, the modem does not start dialing and nothing is written to the log.

Solution: Increase pppd's debugging level to 6 (IP message summary). If it now shows the packets, but says 'dial failed,' then PPP is waiting after a failed call attempt until the call retry delay timer expires. Use SIGINT to tell pppd to reset the timer, change the `Systems` file entry if it is set to use too long a delay, or wait until the timer expires.

Scenario: To avoid using a separate subnet, you set up your network as described in "ARP Table Manipulation" in the section "Connecting a Host to a LAN." The remote workstations can exchange packets with everything on that LAN, but not with any hosts or networks that are reachable through a particular router.

Solution: Some routers implement a security feature that causes them to not believe the second IP address that is claimed to be associated with a particular Ethernet address. If your remote workstation can exchange packets with all the hosts on your LAN but not with any that are reachable only through a particular router, then that router is probably skeptical of your ARP entry. This router security feature is usually configurable. If it is not configurable, you must put the remote workstations on their own subnet as described in "Separate Network" in the section "Connecting a Host to a LAN," and configure the router so that it knows that your office hub is the gateway to that subnet.

Scenario: PPP will not connect and the log file shows 'IPCP: Received bad Configure-Reject.'

Solution: Restart pppd with the novjcomp option. If this fixes the problem, then the peer may be using a buggy IPCP option negotiation algorithm. Try running pppd with the rfc1172-vj or the rfc1172-typo-vj option. If these options do not work, go back to novjcomp and complain to the vendor of the peer's implementation of PPP.

Scenario: PPP will not come up, and the peer complains that it has received a request for IPCP configuration option 3.

Solution: Restart pppd with the rfc1171-addresses option, and report a bug to the vendor of the peer's implementation of PPP. If the peer does not recognize IPCP configuration option 3, it should respond with an IPCP Configure-Reject rather than exiting altogether.

Scenario: The PPP link comes up and carries IP traffic, but a minute later the log file shows 'LQM: LQRs: 0/5,' then 'LQM: Too many Link-Quality-Reports lost,' then pppd closes the connection.

Solution: Restart pppd with either the echolqm or the nolqm option and report a bug to the vendor of the peer's implementation of PPP.

If the peer does not support Link Quality Monitoring, it should issue a Configure-Reject during LCP negotiations. If, even after sending an LCP Configure-Ack in response to HP-UX's PPP LCP Configure-Request containing LQM, the peer does not recognize a Link Quality Report, it should respond with a Protocol-Reject rather than not responding at all. When pppd sees the Protocol-Reject, it will fall back to using LCP Echo-Requests for its link status monitoring functions.

An LQM failure one minute after connection startup, particularly during successful user data transfers, indicates that the peer neither properly Configure-Rejected LQM during LCP negotiations, nor properly Protocol-Rejected HP-UX PPP's first LQR.

Scenario: When the modem abruptly gets hung up, pppd does not notice.

Solution: Make sure that the modem drops the Carrier Detect (CD), also called Data Carrier Detect (DCD), signal when the carrier is lost. On a Telebit T1600, set 'AC1."

Make sure that the cable properly connects CD (RS-232 pin 8) from the modem to the serial port, and that the pppd command line does not include the ignore-cd option.

Scenario: Shortly after the Systems chat script completes, the log file records a SIGHUP and the connection drops. But the remote modem did not actually hang up the connection and tip works correctly.

Solution: Make sure that the modem asserts Carrier Detect (CD), also called Data Carrier Detect (DCD), signal when it is actually talking to a remote modem. On a Telebit T1600, set 'AC1."

Make sure that the cable properly connects CD (RS-232 pin 8) from the modem through to the serial port. pppd does not check the state of DCD until after the Systems chat script completes, about the time that PPP negotiations begin. If the modem is not asserting DCD, or if the cable is not delivering the signal to the serial port, pppd will not be able to tell that the modem is connected. To test this, or to work around a defective cable, start pppd with the ignore-cd option.

Scenario: pppd will not hang up the phone, even though it was started with idle 300.

Solution: Increase pppd's debugging level to 6 (IP frame summary) and watch for logged frames that do not say '!keepup.' If you do not want those messages to keep your line active, put them in the 'keepup' filter in the Filter file ($PPPHOME/Filter by default).

Scenario: pppd dies with a 'Fatal error: ...' or 'Fatal system error: ...' message.

Solution: Contact your HP support representative.

Scenario: When you telnet to test if the answering PPP account is set up correctly, you are greeted with what looks like line noise. This is the PPP Link control Protocol Configure-Request packet, but it takes a long time for pppd to give up and go away.

Solution: The easy way to instruct the answering pppd to exit and close the connection is to type the four-character sequence of a tilde (~), followed by three control-Cs.

The '~^C^C^C' sequence only works in PPP mode, not in SLIP mode. And in PPP mode, if you specified 'passive' on the command line, '~^C^C^C' only works after receipt of a good PPP packet.

Scenario: pppd will not dial out; the log file says 'Device is locked.'

Solution: pppd uses lock files compatible with other programs to avoid using a device already in use by another program, and to keep other programs from interfering with devices in use by pppd. If the pppd.log file says 'Device is locked,' look for a lock file such as LCK..cua in /usr/spool/locks, /usr/spool/uucp, /usr/spool/uucp/LCK, /var/spool/locks, or wherever your system keeps tty lock files. Each lock file should contain the process ID of the owner process. If the lock file is four bytes long, the process ID is a 32-bit binary number. If the lock file is 11 bytes long, the process ID is printed in ASCII. Look in the lock file for the owner PID (use od -d for binary files and cat for ASCII lock files), and then use ps to look for the owner.

When pppd finds a lock file of zero length, it has no way of knowing the PID of the process that created it, so to be safe, it gives up trying to open the associated tty.

Scenario: Even though UUCP is using the modem, pppd tries to use it to place an outbound call. And when pppd has an active link, UUCP tries to make an outbound call.

Solution: The outbound 'Auto Call Unit' (ACU) names used in PPP's Devices must match the names used for similar purposes in UUCP's Devices file. For example, if your UUCP configuration knows a device as cua0, but PPP knows it as cua, each facility will be unable to discover the lock file indicating that the other is using the device.

Scenario: With the debugging verbosity level set to 2 or more, messages appear in the log file like 'Bad FCS received,' 'Bad protocol (even), flushing frame,' 'Short frame received (3 bytes),' 'Frame too long, flushing frame,' 'Missed ALLSTATIONS, flushing frame,' or 'Missed UI, flushing frame.'

Solution: Some of the incoming messages are getting damaged in transit. If your throughput is erratic or lower than you expect, then refer to the section on "PPP connects and everything works, but it is slow and erratic."

If you see 'Missed ALLSTATIONS' while the link is coming up, after 'Chat script succeeded' but before the first 'Received LCP configure-Request,' do not worry. The calling daemon is scanning the characters it sees coming from the answering system, looking for the start of a PPP frame that signifies the beginning of option negotiations. While the contents of the answering machine's /etc/motd scrolls by, it is likely that the calling daemon will flush several of what it had thought to be frames, but that turned out to be more text.

If the messages happen while the link is active, but only rarely, then do not worry about them. The errors may be due to intermittent line noise in the phone lines, or your computer may occasionally lose incoming characters. This last possibility is likely if the errors tend to occur when receiving large amounts of data.

Scenario: /etc/resolv.conf points at a name server out on the Internet someplace, across the PPP link. When /etc/resolv.conf is in place and the PPP connection is up, connectivity between internal hosts is fine. When /etc/resolv.conf is in place and the PPP connection is down, connectivity between internal hosts slows down significantly. Once the /etc/resolv.conf is removed, connectivity returns to normal.

Solution: Local applications are trying to reverse-resolve incoming connection addresses (even on the local network) into names, whether for authentication (for example, /.rhosts and /etc/hosts.equiv) or just normal operations (for example, telnetd making entries into /etc/utmp and /var/adm/wtmp). When the DNS resolve routines cannot reach the name server, they do not return a value that means "I don't know" until some timeout interval has passed. This is probably happening on every connection attempt.

Run a local name server that is either Start of Authority (SOA) or an authoritative secondary Name Server (NS) for your name space (forward mapping) and your address space (reverse mapping). An authoritative secondary NS is one that is listed in the SOA's Resource Record (RR) for that domain.

Scenario: Using PPP on a system or network where Frame (a WYSIWYG document production system) is installed, pppd dials the modem hourly even though no users are actively using it. The pppd.log file shows a line like:

```
9/23-18:48:34-83 udp 192.0.2.1/3680 ->
192.0.2.2/sunrpc 45 bringup
```

Solution: This is Frame's copy license manager probing for other license managers on the network. To avoid bringing up the link for this sort of packet, append '!sunrpc' to the 'bringup' filter in the Filter file.

Scenario When PPP is used in SLIP mode, the peer receives only alternating frames. Every other frame is discarded.

Solution The peer's SLIP implements only the optional framing style suggested in the PROTOCOL section of RFC 1055, and cannot receive adjacent frames separated by only one END character (0xC0). Add extra-slip-end to the pppd command line.

A Modem Connections

This appendix includes some basic information about types of transmissions, proper cabling between serial ports, and modem configuration. The information is general and is not a substitute for

HP-UX system and modem documentation. The system and modem manuals should take precedence if you have any questions about configuration and equipment.

RS-232 Interface

The types of signals carried by the RS-232 interface are listed below:

Signal	Description
Signal Ground (SG)	Common electrical ground path for all interface circuits.
Transmit Data (TD)	Command codes or data sent from DTE to DCE; won't be sent unless RTS, CTS, DSR and DTR are set to on.
Receive Data (RD)	DCE responses to DTE commands or data received from a remote DCE.
Request To Send (RTS)	Set to on, the DTE tells the DCE to remain in transmit mode; set to off, the DTE tells the DCE to receive.
Clear To Send (CTS)	Set to on, the DCE reports it is ready to receive; set to off, the DCE tells the DTE not to transmit.
Data Set Ready (DSR)	Set to on, the DCE reports it is ready to operate.
Data Terminal Ready (DTR)	Set to on, the DTE tells the DCE to connect to a communications channel (required to be on for automatic answering by DCE). Set to off, the DTE tells the DCE to break the connection when it has finished the current transmission.
Ring Indicator (RI)	Set to on, the DCE reports it detects the presence of a ringing signal on the communications channel. Set to off, the DCE indicates the absence of a ringing signal.
Carrier Detect (CD)	Set to on, the DCE reports receiving a signal from the communications channel that a connection can be established.

The RS-232 standard utilizes nine pins for signaling, although many serial interfaces provide 25. The other 16 pins can be used for testing or secondary signaling. Smaller 9-pin connectors support standard signals and save space.

DB-25

25 pin connectors, or DB-25, use pins 2 though 8 and pins 20 and 22 as shown:

Pin	Signal	Signal Direction
1	Protective Ground	both
2	Transmit Data (TD)	from DTE
3	Receive Data (RD)	from DCE
4	Request To Send (RTS)	from DTE
5	Clear To Send (CTS)	from DCE
6	Data Set Ready (DSR)	from DCE
7	Signal Ground	both
8	Carrier Detect (CD)	from DCE
20	Data Terminal Ready (DTR)	from DTE
22	Ring Indicator (RI)	from DCE

DB-9

Standard 9 pin connectors, called DB-9, use the following pin setup:

Pin	Signal	Signal Direction
1	Carrier Detect (CD)	from DCE
2	Receive Data (RD)	from DCE
3	Transmit Data (TD)	from DTE
4	Data Terminal Ready (DTR)	from DTE
5	Signal Ground	both
6	Data Set Ready (DSR)	from DCE
7	Request To Send (RTS)	from DTE
8	Clear To Send (CTS)	from DCE
9	Ring Indicator (RI)	from DCE

DB-9 to DB-25 Conversion

This wiring diagram shows the standard conversion from DB-9 to DB-25:

Pin	DB-9 Pin	DB-25 Pin
Carrier Detect (CD)	1	8
Receive Data (RD)	2	3
Transmit Data (TD)	3	2
Data Terminal Ready (DTR)	4	20
Ground	5	7
Data Set Ready (DSR)	6	6
Request To Send (RTS)	7	4
Clear To Send (CTS)	8	5
Ring Indicator (RI)	9	22

HP Modem Cables

Most HP systems do not use straight through modem cables to connect the modem to the system. Make sure you are using the correct cable that is proper for your system.

HP Modem Cable (Series 700)

The HP 9000/700 series workstation provides a DB-9 port for connection to aysnchronous serial devices. Hewlett-Packard recommends that you use cable part number 24542M to connect a 9000/700 system to a DB-25 asynchronous modem. A similar part number (24542G) describes a printer cable. Do not confuse the two part numbers if you order the Hewlett-Packard cable. If you cannot get the HP part, use a cable like the one described in the table below.

Signal	Direction	DB-9 Pin	DB-25 Pin
Carrier Detect (CD)	from Modem	1	8
Receive Data (RD)	from Modem	2	3
Transmit Data (TD)	from DTE	3	2
Data Terminal Ready (DTR)	from DTE	4	20
Ground	both	5	7
Data Set Ready (DSR)	from Modem	6	6
Request To Send (RTS)	from DTE	7	4
Clear To Send (CTS)	from Modem	8	5
Ring Indicator (RI)	from Modem	9	22

HP Modem Cable (Series 800)

For an 800 series system with a model 5062-3054 MDP "Full Modem" connector panel use the HP part #40233A modem cable. If you cannot get the HP part, use a cable like the one described in the table below.

CPU		DIRECTION	MODEM	
Gnd	1	both	1	Gnd
TD	2	from Modem	3	RD
RD	3	from DTE	2	TD
RTS	4	from Modem	8	DCD
DSR	6	from DTE	20	DTR
GND	7	BOTH	7	GND
DCD	8	from DTE	4	RTS
	9	from Modem	22	RI
DTR	20	from Modem	6	DSR
RI	22	from Modem	5	CTS

Null-Modem Cables

You can directly connect the serial ports of two systems with a null-modem cable. Null-modem cables connect pins of one machine to their symmetric counterparts on another machine. For example, pin 2, the Transmit Data (TD) pin of machine A, is paired with the pin 3, the Receive Data (RD) of machine B. If this is not done, both machines would try to use pin 2 to transmit data and neither could receive the others transmission. Sometimes only pins 2, 3, and 7 are connected in a null-modem cable, but hardware handshaking may require optional connections of other pins. Data Set Ready (DSR) and Carrier Detect (CD), pins 6 and 8, may be joined together and connect to the other machine's Data Terminal Ready pin, number 20.

The table below shows typical null-modem pin connections between two DB-25 ports.

DB-25 to DB-25 Null-Modem Connections

DTE Signal	DTE Pin	DCE Pin	DCE Signal
Protective Ground	1	1	Protective Ground
Ground	7	7	Ground
Transmit Data (TD)	2	3	Receive Data (RD)
Receive Data (RD)	3	2	Transmit Data (TD)
Data Set Ready (DSR) & Carrier Detect (CD)	6+8	20	Data Terminal Ready (DTR)
Data Terminal Ready (DTR	20	6+8	Data Set Ready (DSR) & Carrier Detect (CD)
Request To Send (RTS)	4	5	Clear To Send (CTS)
Clear To Send (CTS)	5	4	Request To Send (RTS)

Dial Up Modems

PPP works well with any number of brand-name, non-proprietary, dial-up modems. Modems for dial up protocols like PPP should conform to non-proprietary standards because the local user will probably have little knowledge of the equipment at the remote end of the connection. A non-proprietary modem's ability to match carrier speed will make a usable connection with whatever type of modem is operating at the other end of a connection.

This section discusses a few features of modem performance and configuration, including speed of transmission, error correction, data compression, flow control, command mode, and S registers.

Speed

The higher the speed associated with the modem, the faster the data transmits. There are two ways to gauge modem speed. One figure, usually between 9600 and 28800 bps, measures the amount of data that can be transmitted or received between modems. The other, a much higher figure, perhaps 38400 to 115200 bps, describes data throughput between a system and its modem. When you configure PPP, choose the highest modem throughput that can be supported by the system. Typically, this speed will be 38400. The figure can be found in the files `/usr/include/sys/ttydev.h` or `/usr/include/sys/termio.h`.

In general, the supported baud rate is limited by the system hardware. For example, some serial interface cards support a maximum speed of 19200 baud whereas series 700 built-in serial ports support a maximum speed of 57600 baud. `pppd` supports up to 57600 baud in the software if the hardware can support this speed.

Data Compression

Enable data compression if your modem supports it, unless your application performs better without it. Data compression maximizes the amount of information crossing a PPP connection. Generally, executable (binary) files do not compress well and files that have been compressed before transmission are difficult to compress further, but most protocols provide at least 2:1 compression ratios for other types of data. Microcom, Inc.'s Microcom Networking Protocol 5 (MNP 5) protocol can double the

data transmitted, and MNP 7 offers 3:1 compression. CCITT-sanctioned V.42bis can compress data 4 to 1 under the best conditions. If your modem offers a choice between an MNP protocol and V.42bis, choose the latter. In addition to having a better compression algorithm, it works better with precompressed data streams.

Run the serial port at its maximum speed (usually 38400 bps) to gain the most benefit from in-modem data compression. Some modem's data compression adds significant latency to data throughput, adversely affecting interactive responsiveness between local and remote users. Experiment with your modem and the type of data you transmit to find the best configuration.

PPP supports several other kinds of compression that work at different layers of the communications stack. For more information on the following types of compression supported by PPP, see `pppd(1`:

- HDLC Frame
- Address, Control and Protocol Fields
- Van Jacobson TCP Header
- PPP Link
- Predictor-1

Error Correction

In addition to data compression protocols, Microcom, Inc. developed error correction protocols with similar names. Its MNP 4 is one of two major protocols for error correction. The other is Link Access Procedure for Modems (LAPM). Many modems support either one of the two protocols or both protocols. Some manufacturers have developed their own proprietary error control protocols, but the CCITT V.42 standard requires that a modem support MNP4 and/or LAPM to be compliant or compatible with the standard. The receiving and transmitting modems negotiate to discover the highest level of error correction protocol each can support.

Error correction verifies data that has traveled across the communications connection. Noisy lines destroy data in transmission, especially when the transmission is high speed. Depending on the protocol, bits, bytes or packets are subjected to some form of cyclical redundancy check to ensure the integrity of the received data. The receiving end informs the transmitting end when the data has been

damaged. The protocol used determines whether just the corrupted data, or all data sent since the error was discovered, is transmitted again. MNP 4 and later protocols can also respond to poor transmission quality by causing the data to be shipped in smaller packets, increasing the number of checks and decreasing the amount of data corrupted by line bursts.

Flow Control

Flow control provides the modem with a buffer for storing received data from the system. This is beneficial because data transmission between the modem and system or between two systems connected by a null-modem cable is much faster than transmission between modems. Flow control keeps system-to-modem data from being lost during the modem-to-modem transmission.

NOTE The computer and modem must use the same type of flow control.

Flow control can be provided by system software or modem hardware. Although `pppd`'s default status is no flow control, you should use hardware or software flow control if your system supports flow control. Hardware flow control is recommended.

Hardware flow control is managed by the Request To Send (RTS) and Clear To Send (CTS) signals. If you use hardware control, make sure the connecting cable carries both signals. Also make sure the modem is configured for hardware flow control. Lastly, on a HP-UX system, you must configure the device to use hardware flow control. Instructions on how to do this are later in the section “Enabling Hardware Flow Control.”

If both your serial interface card and your modem support hardware flow control (RTS/CTS), HP recommends using it. The minor number of the device file has a bit to indicate hardware flow control. All systems with HP Precision Bus (HP-PB) architecture support hardware flow control. Refer to `termio(7)` for specific models and references.

If you choose to use software flow control, then make sure the modem is configured for software flow control. You should also instruct `pppd` to use software flow control by using the option xonxoff either in the Devices file or on the `pppd` command line.

If you configure for software control, make sure both ends of the PPP connection have the 0x000A0000 bits turned on in their asynchronous control character map (asyncmap). This is the default for PPP. A computer and modem using software flow control can talk to a computer and modem using hardware flow control, but both ends must have the asyncmap set to accommodate software flow control.

Modem Commands

Most modems are intelligent and recognize commands sent to them by the system during DTE to DCE transmission. Commands are different than signals exchanged by the DTE and DCE, like RTS and CTS. Generally, signals indicate the state of the DTE or DCE. Commands tell the modem to perform special tasks or modify dialing procedures. Common commands cause the modem to answer an incoming call, use touch tone dialing, terminate the current connection, or pause before sending the next number in a string. The modem recognizes system commands because they are preceded by the characters AT. Therefore, the commands are referred to as the AT command set.

Common Commands	Description
A	Answer incoming call; overrides S0 register.
E*n*	Echo command; E0 disables echo to monitor; E1 enables echo.
H*n*	Switch hook control; H0 hangs up; H1 goes on-line.
O	Switch from command mode to data mode.
X*n*	Select call progress code set.
&C*n*	Select Carrier Detect option.
&F*n*	Load factory setting number n.
&V	List modem's stored configuration files.

Common Commands	Description
&W*n*	Save current modem settings as n.
\N*n*	Select error control mode.
\Q*n*	Select serial port flow control.

S Registers

S registers are settings stored in the modem's memory. The S register format is S*r*=*n* where *r* is the register's number and *n* is the register's value. For example, S0=1 indicates the modem should answer an incoming call after the first ring. S7=30 tells the modem to wait 30 seconds after dialing before establishing a connection.

There are 256 possible S registers, although some are reserved and many are not used at all. Each manufacturer has a proprietary license to use S registers as it likes, but some registers are fairly consistent. Some are shown in the table below, although there is no guarantee they will match your modem's settings. Review your system or modem documentation to determine the meanings and values of your modem's S registers.

S Register	Description
S0	In auto answer mode, which ring to answer on
S1	Ring counter
S2	Escape character; default is +
S3	Carriage return character; default ASCII 13
S4	Line feed character; default is ASCII 10
S5	Backspace character; default is ASCII 18
S7	Seconds to wait after dialing to establish a connection
S9	Tenths of seconds to wait before issuing carrier detect signal
S46	Select error control and data compression

Common Modem Configurations

In the next section you will find some suggestions for Telebit modem settings which have worked well during testing. The examples here show the complete register settings for a Telebit T1600 and a description of the settings. Following this information are some command strings that will provide the proper register settings for Telebit's T1600, Qblazer and T3000 modems. Most modems should provide equivalent settings. Check your modem's user guide for help in composing the command string for similar settings.

Telebit Modem Settings

The following commands were used to configure settings for a trio of Telebit modems:

```
T1600: at &f s0=1s7=120s48=1s51=6s58=2s59=15s61=0s63=2tq2x12&c1&d3&s1 &w
T3000: at &f9 s0=1s7=120s48=1s51=6s58=2s59=15s61=0s63=1tq2x12&c1&d3&s1 &w
Qblazer: at &f s0=1s7=120s48=1s51=6s58=2s59=15s61=0s63=2tq0x4&c1&d3&s1 &w
```

The commands produced the following list of settings for the Telebit T1600. The list is the output from the command AT&V.

```
at&v
T1600 - Version LA1.00 -Active Configuration
    a1 E1 L0 M0 Q2 T V1 X12 Y0
 &C1 &D3 &G0 &J0 &L0 &Q0 &R3 &S1 &T4 &X0

S000:1 S001=0 S002=43 S003=13 S004=10 S005=8 S006=2 S007:120
S008=2 S009=6 S010=14 S011=70 S012=50 S018=0 S025=5 S026=1
S038=0 S041=0 S045=0 S046=0 S047=4 S048:1 S050=0 S051:6
S056=17 S057=19 S058:2 S059:15 S060=0 S061:0 S062=15 S063:2
S064=0 S068=255 S069=0 S090=0 S093=8 S094=1 S100=0 S102=0
S104=0 S105=1 S111=255 S112=1 S180=2 S181=1 S183=25 S190=1
S253=10 S254=255 S255=255 OK
```

The table below shows the features and S register values set by the T1600 commands. Many of these settings, particularly the S registers, only apply to Telebit modems. However, your own modem will likely respond to a different command or register value in the same way the T1600 does to these. Use your modem's command set and register semantics to set features like RTS/CTS hardware flow control. Refer to the `Dialers` file for descriptions of several more modem configurations.

Command or setting	Result
&f	Load factory default settings.
s0=1	Answer after one ring.
s7=120	Wait 120 seconds for a valid carrier tone to be sent from the remote modem.
s48=1	Compare all eight bits when checking for control characters.
s51=6	Latch the DTE interface at 38400 bps.
s58=2	Use full-duplex RTS/CTS flow control so the modem sends data to the DTE when RTS is on and will not send data to the DTE when RTS is off.
s59=15	Use the following result code suffixes: Possible link protocol suffixes: nothing, REL, or LAPM. Possible compression suffixes: nothing or COMP.
s61=0	Local reaction to a break signal on the serial interface is the same as specified in S63.
s63=2	Discard buffered data and send the break signal when a break signal is transmitted by the local DTE during an error controlled connection.
q2	Report result codes when originating a call, but do not return result codes when answering a call.
t	Use DTMF tone dialing.
x12	Result codes include the following: BUSY,DIALING, ERROR, NO CARRIER, NO DIALTONE, OK, RING, RRING, CONNECT 300, CONNECT 1200, CONNECT 2400, CONNECT 4800, CONNECT 7200, CONNECT 9600,CONNECT 12000, CONNECT 14400, CONNECT 19200, and CONNECT 38400.

Command or setting	Result
&c1	Turn ON the DCD signal when a remote modem carrier is detected and turn DCD OFF when the carrier is dropped.
&d3	Recall the current user configuration parameters from nonvolatile memory and enter command mode when the DTR signal is switched from ON to OFF.
&s1	Turn DSR ON after the answer tone is detected and leave it ON throughout the connection.
&w	Save the current configuration settings to nonvolatile RAM.

MultiTech and Hayes Smartmodems Settings

The following are general recommendations for configuring your modem. Refer to your modem manual for a detailed explanation of the modem settings.

If you are using a MultiTech V.32B modem, the following register settings are helpful:

```
>atL5

B1 E1 M1 Q0 R0 V1 X0 &E1 &E4 &E6 &E8 &E10 &E13 &E15
$MB9600 $SB9600 $BA0 &W1
```

The following list explains each of these register settings.

B1 selects Wait-for-Dial-Tone Dialing

E1 means do echo Command Mode characters

M1 means Monitor speaker On until Carrier detected

Q0 means Result Codes sent

R0 means modem will not reverse modes

V1 means Result Codes sent as words (verbose mode)

X0 selects Basic Result Codes

&E1 means V.42 Auto-Reliable Mode

&E4 means CTS modem-initiated flow control

&E6 means Xon/Xoff not passed through

&E8 means Enq/Ack pacing off

&E10 means Normal Mode flow control off

&E13 means Pacing on

&E15 means data compression enabled

$MB9600 selects 9600 bps on-line

$SB9600 selects 9600 bps at serial port

$BA0 means Baud Adjust is off, speed conversion is on

&W1 causes modem to store its current parameters into non-volatile RAM, and modem will load these for future sessions instead of reading factory ROM and DIP switch defaults

Enabling Hardware Flow Control

The following example shows a common method for setting hardware flow control in HP-UX 10.x series 700 and 800 machines. However, remember that HP technical support is the best resource for information about setting up hardware flow control.

The example is presented in two phases:

1. Adding devices with the SAM interface.
2. Setting bits in the minor device number to enable hardware flow control.

In this example an outbound device (`/dev/cul0p1`) and an inbound device (`/dev/tty0p1`) are added without hardware flow control. Next, the new devices are renamed and the mknod command is used to set hardware flow control for the devices.

To use this example to set up hardware flow control on your own system, substitute your device names where appropriate. Additional information may be obtained from `termio(7)`.

Adding the devices

1. Use SAM to add the modem devices.

   Choose "Peripheral Devices," then "Terminals and Modems."

2. Write down the names of the devices you create.

Outbound devices begin with the string "cu" and inbound devices begin with the string "tty."

3. Type the following, substituting device names you wrote down in step 2 for the italicized device names in this example:

```
ls -l /dev/cul0p1 /dev/ttyd0p1
```

Output similar to the following appears.

```
crw-rw-rw-   1 bin       bin        1 0x000001 Jul  8 13:51 cul0p1
crw-rw-rw-   1 bin        bin      1 0x000002 Jul  8 13:51 ttyd0p1
```

The output contains important device configuration information.

```
crw--w----   1 bin       bin     1 0x000002 Jul  8 13:51 ttyd0p1
                                 ^       ^^
                                 |       ||
                                 |       ||
                               major     |+indicates inbound or outbound
                              number     | device
                                         |     1 = outbound, 2 = inbound
                                         +-indicates status of hardware
                                           flow control
                                              0 = off, 1 = on
```

4. Type the entry below, inserting the major number as shown. In this example, the major number is 1.

```
lsdev 1
```

Output similar to the following appears.

```
Character     Block       Driver          Class
   1            -1           asio0             tty
```

5. Proceed to “Setting Hardware Flow Control” if the name “tty” appears under the heading, “Class”.

NOTE Do not proceed if the name under the heading “Class” is "pseudo". If “pseudo” appears, the tty driver is assigned a (potentially) new major number each time the system is rebooted. The next section only applies if your system's major numbers are static, which is usually the case. If your system does not have static major numbers, you will need to contact HP technical support for further instructions on how to set up hardware flow control for your tty devices.

Setting Hardware Flow Control

The output in step 3 of "Adding the Devices" shows that neither device has hardware flow control set. This is indicated by the zero (0) in the next to last place of the minor device number in each line. This example shows how to turn on hardware flow control by creating new devices with the proper minor mode bits set to one (1).

1. Write down the following information before proceeding to the first step of setting the hardware flow control:
   - the major number (1 in the above example)
   - the minor numbers (0x000001 and 0x000002 in the above example)
2. Rename the devices by issuing the following commands. This saves your original device configurations under new names. If you ever need to revert to devices that do not have hardware flow control set, you can move the original files back into place.

   ```
   mv /dev/cul0p1 /dev/cul0p1.orig
   mv /dev/ttyd0p1 /dev/ttyd0p1.orig
   ```

3. Use the `mknod(1)` command, as shown, to create new devices with hardware flow control, changing only the hardware flow control bit from "0" to "1".

   ```
   mknod /dev/cul0p1 c 1 0x000011
   mknod /dev/ttyd0p1 c 1 0x000012
   ```

4. Type the following and check your work by examining the output.

   ```
   ls -l /dev/cul0p1 /dev/ttyd0p1

   crw-rw-rw-   1 root   sys  1   0x000011 Jul  8 14:09 cul0p1
   crw-rw-rw-   1 root   sys  1   0x000012 Jul  8 14:09 ttyd0p1
   ```

Notice that the hardware flow control bit is now set to "1". Make sure your modem is set up for hardware flow control as well. For closure, set the permissions and ownership to be the same as they were when the devices were first created via SAM.

Some modems may refuse to answer the phone if RTS recognition is enabled and RTS from the HP is low. Even when `uugetty` is used to enable incoming calls on the port, these modems will not answer the call because `uugetty` does not assert the RTS signal into the modem. The solution is to turn off RTS recognition on the modem. This is usually

done with the modem command AT&R1. Unfortunately, hardware flow control will no longer work reliably on this port. If you encounter problems, you should consider using software flow control instead.

Glossary

Active-Open: an event internal to the PPP configuration finite state machine that causes PPP to start sending Configure-Request messages to the peer.

bps: bits per second.

Challenge Handshake Authentication Protocol (CHAP): a challenge-response LCP authentication protocol resistant to playback attacks. CHAP runs after LCP negotiation is complete but before any NCPs are started.

CHAP: *see Challenge Handshake Authentication Protocol.*

DNS: see Domain Name System.

Domain Name System (DNS): the distributed host and network name database system used to translate between Internet addresses and their associated host and network names. RFC 1034 is an introduction to DNS. RFC 1035 describes the protocols and data types used.

FCS: *see Frame Check Sequence.*

File Transfer Protocol (FTP): internet protocol, described in RFC 959, used to copy data files across TCP/IP networks.

Frame Check Sequence (FCS): the 16-bit field transmitted after all data in a frame but before the closing flag. Its value is computed using the data portion of the frame and a binary polynomial, and its purpose is to provide the receiver with a fairly reliable check against data corruption during transmission. An 'FCS error' indicates that a frame was damaged in transit.

FTP: *see File Transfer Protocol.*

HDLC: hardware flow control: out of band flow control defined in EIA RS-232-D that provides bidirectional flow control using the Request to Send (RTS) and Clear to Send (CTS) signals. The DTE (typically a computer) asserts RTS when it is able to accept input, and the DCE (typically a modem) asserts CTS when it is able to accept input.

High-Level Data Link Control (HDLC): a bit-oriented framing and addresing scheme defined in ISO 3309 that is used as the basis for most modern synchronous wide-area data communications protocols, including PPP, X.25, SNA, OSI, and Frame Relay.

Glossary

International Standards Organization (ISO): organization that publishes a broad range of international standards for industry, including a large number that are identical or nearly identical to CCITT standards.

Internet Protocol (IP): the layer of the Internet family of protocols that is responsible for packet routing and datagram fragmentation and reassembly.

Internet Protocol Control Protocol (IPCP): the Network Control Protocol for the Internet Protocol.

IP: *see Internet Protocol.*

IPCP: *see Internet Protocol Control Protocol.*

ISO: *see International Standards Organization.*

LCP: *see Link Control Protocol.*

Link Control Protocol (LCP): the protocol that establishes, configures, and tests the data link connection.

Link Quality Monitoring (LQM): a negotiable feature of LCP that allows a PPP implementation to determine when and how often the link is losing data. If both ends of a PPP link agree to perform LQM, they will periodically send Link Quality Report messages to their peer containing various sequence numbers, state variables, and byte and packet counts. This information is then used to measure how reliable the connection is, and can be used to decide when to shut down the connection down and perhaps try again.

Link Quality Report (LQR): an LCP message sent after successfully negotiating the LQM option. The LQR contains various state variables, packet counts, and byte counts.

LQM: *see Link Quality Monitoring.*

LQR: *see Link Quality Report.*

Magic Number: a 32-bit random number, unique to each end of each PPP link, used to determine if the link is looped back. If a PPP implementation discovers that its peer has the same Magic Number, it will strongly suspect that it is talking to itself.

Glossary

Maximum Receive Unit (MRU): the size of the data field in the largest PPP message a PPP implementation can receive.

Maximum Transmission Unit (MTU): the size of the largest IP datagram an IP interface can send.

MRU: *see Maximum Receive Unit.*

MTU: *see Maximum Transmission Unit.*

NCP: *see Network Control Protocol.*

Network Control Protocol (NCP): any of a group of protocols that run after LCP has successfully connected and whose purpose is to establish and configure a network-layer protocol.

Network-Layer Protocol: the member of any protocol family corresponding to Level 3 on the ISO protocol stack. One example is IP.

Packetized Ensemble Protocol (PEP): a proprietary modulation and error correction technique used in some Telebit modems. It can achieve throughput of up to 16000 bps, and is able to establish and maintain connections over noisy lines better than V.32, but is is asymmetric, with slow line turnaround and latency.

PAP: *see Password Authentication Protocol.*

Passive-Open: an event internal to the PPP configuration finite state machine that causes PPP to move from the Closed state to the Listen state, where it awaits Configure-Request messages from the peer.

Password Authentication Protocol (PAP): a simple LCP authentication protocol that sends an identifying name and an associated password. PAP runs after LCP negotitation is complete but before any NCPs are started.

peer: the PPP implementation at the other (far) end of the link.

PEP: *see Packetized Ensemble Protocol.*

Point-to-Point Protocol (PPP): the Internet standards-track data-link protocol for carrying multi-protocol datagrams (including IP) over serial links.

PPP: *see Point-to-Point Protocol.*

Glossary

SCSI: *see Small Computer System Interface.*

Serial Line Internet Protocol (SLIP): a simple protocol for carrying IP datagrams over serial links. No error detection or configuration negotiation is included.

SLIP: *see Serial Line Internet Protocol.*

Small Computer System Interface (SCSI): a standard for connecting high-speed intelligent peripherals, such as disk or tape drives, to computers.

telnet: the protocol used to establish terminal sessions across internet networks. See RFC 854 for the protocol specification. There are many other related RFCs that describe various telnet options.

V.32: a 9000 bps full-duplex modem modulation scheme with fallback to 4800 bps.

V.32bis: a 14400 bpx full-duplex modem modulation scheme with fallback to 12000, 9600, 7200, and 4800 bps.

V.42: an error correction scheme used by many V.32 and V.32bis modems that carries asynchronous data using the LAPM protocol over a synchronous connection.

V.42bis: a data compression method used by many V.32 and V.32bis modems. V.42bis can only be used over V.42.

Index

Symbols

/etc/passwd, 24, 37
/etc/ppl/ppl.remotes, 61
/etc/ppp/Autostart, 24, 33, 35
/etc/ppp/Devices, 24, 33, 41
/etc/ppp/Dialers, 24, 33
/etc/ppp/Login, 24
/etc/ppp/ppl.remotes, 47
/etc/ppp/Systems, 24, 33, 34, 41
/etc/uucp/Devices, 47
/etc/uucp/Dialers, 47
/etc/uucp/Systems, 47

A

Active negotiation, 65
Address compression, 75
ARP table manipulation, 83
Automatic failover option, 79

C

Challenge Handshake Authentication Protocol (CHAP), 133
Closed policy filter example, 119
Command line
  IP addresses, 42
Command line options, 63
  compression, 75
  dedicated lines, 78
  exec, 68
  idle timer, 66
  link management, 65
  LQM, 72
  negotiations, 65
  SLIP, 70
  synchronous mode, 78
Common problems, 135
Compression
  address and control field, 75
  HDLC frame, 75
  PPP link, 76
  Predictor-1, 77
  protocol field, 75
  Stac LZS, 77
  TCP header, 76
  Van Jacobson, 76
Compression option, 75
Configuration
  additional information, 40
  creating device files, 28
  files, 23
  filter file, 40
  inbound connections, 37
  IP addresses on command line, 42
  IP tunnets, 30
  matching search criteria, 41
  modem, 31
  non-generic login script, 40
  outbound connections, 33
  overview, 23
  quick reference, 26
  steps, 26
  testing connections, 39
Connection establishment, 24
Connections
  testing, 39
Constantly-open telephone calls, 80
Control field compression, 75

D

Data compression, 70
Definition of PPP, 16
Device files
  creating, 28
Dial-back, 131
Dialing in to HP 9000, 17
Dialing out from HP 9000, 18
Direct connections to HP 9000, 18

E

Establishment of connection, 24
Example connection, 24
exec option, 68

F

Files for PPP, 21
Filter file, 40, 89
  rulesets, 90
Filters, 91
  closed policy example, 119
  open policy example, 125
  stanzas, 92

G

getty, 134

H

HDLC frame compression, 75
HP 9000
  dialing in, 17
  dialing out, 18
  direct connections, 18

I

ICMP packets, 94
Idle timer option, 66
Inbound connections
  configuring, 26, 37
Installation of PPP, 21
Introduction to PPP, 16
IP addresses
  dynamic address assignment, 42
  list selection, 42
  soft addresses, 42
  tty name calculation, 43
IP addresses on command line, 42
IP packets, 93
IP routing, 81
IP tunnels, 30

Index

L

LAN connections via PPP, 81
LAN/9000, 16
Link management, 65
Link peer authentication, 133
Link Quality
  Monitoring.SeeLQM.
Login shell script, 17, 37
  non-generic, 40
LQM
  adjusting default, 72
  link failures, 73
  peer refusal, 74
LQM option, 72

M

Migrating SLIP to PPP
  connections, 45
Modem
  configuring for PPP, 31

N

Negotiations
  inbound, 65
  outbound, 65

O

Open policy filter example, 125
Outbound connections
  configuring, 26, 33
Overview of PPP, 17

P

Packets
  ICMP, 94
  IP, 93
  TCP, 95
  UDP, 96
Passive negotiation, 65
Password Authentication
  Protocol (PAP), 133
Point-to-Point Link. See PPL.
Point-to-Point Protocol. See PPP

PPL, 20
ppl.ipool file, 60
ppl.users file, 60
PPP
  automatic failover, 79
  command line options, 63
  compression option, 75
  configuration steps, 26
  connections, 24
  constantly-open telephone
    calls, 80
  dedicated lines, 78
  definition, 16
  exec option, 68
  features, 18
  idle timer option, 66
  installation, 21
  introduction, 16
  IP routing, 81
  LAN connections, 81
  LQM option, 72
  negotiations, 65
  product overview, 17
  security, 87
  SLIP option, 20, 70
  synchronous mode, 78
PPP link compression, 76
pppd, 17, 19, 24
  command line options, 63
  compression option, 75
  exec option, 68
  IP addresses on command line,
    42
  link management options, 65
  Link Quality Monitoring
    option, 72
  replacing getty, 134
  SLIP option, 70
  troubleshooting, 135
ppplstat command, 59
PPP-RUN fileset, 21
Predictor-1 compression, 77
Problems
  common, 135
Product overview, 17
Protocol field compression, 75

R

Replacing getty, 134
Rulesets
  design, 90
  filters, 91
  order, 90

S

SAM, 28
Security, 87
  dial-back, 131
  filter file, 89
  link peer authentication, 133
  replacing getty with pppd, 134
  time-to-call restrictions, 130
Security policies, 89
Serial port device files, 28
SLIP, 20
  configuration files (pre-HP-UX
    10.30), 47
  connection restrictions, 20
  data compression, 70
  migrating to PPP, 45
  migration examples, 50
  option for connection, 70
Stac LZS compression, 77
Stanzas, 92
  all keyword, 104
  frag keyword, 103
  ICMP messages, 100
  IP addresses, 99
  IP option keywords, 103
  IP protocol keywords, 99
  log keyword, 106
  netmasks, 99

Index

origin and destination keywords, 101
port numbers, 99
port numbers and services, 100
syntax, 107
TCP examples, 114
TCP packet header keywords, 102
time-based keywords, 104
trace keyword, 106
UDP example, 108
unreach keyword, 104
writing, 97, 98
Steps in configuring connections, 26
System Administration Manager. See SAM.

T

TCP example stanzas, 114
TCP header compression, 76
TCP packets, 95
Testing connections, 39
Time-to-call restrictions, 130
Troubleshooting, 135

U

UDP example stanza, 108
UDP packets, 96

V

Van Jacobson compression, 76