IFHP-HOWTO

29 Oct 2000 (For ifhp-3.4.2)

Patrick A Powell

papowell@astart.com
AStArt Technologies addr
9475 Chesapeake Dr, Suite D,
San Diego, CA 92123
Phone 858-874-6543
Fax 858-279-8424

IFHP-HOWTO: 29 Oct 2000 (For ifhp-3.4.2)

by Patrick A Powell

The **ifhp** program is an enhanced, extended, highly configurable, and portable implementation of a print filter for use with the **LPRng** Print spooler package. **`ifhp`** supports network, serial, and parallel printers, does page accounting and job recovery, and allows an extremely high level of configuration and tuning. **`ifhp`** gets its flexibility by using a configuration file to set its operational characteristics. The configuration file can contain multiple separate printer configurations and the configuration selection is done by a very simple command line option. The filter supports text, PostScript, PCL, and PJL printers, and can be configured to handle a wide range of printer quirks and mis-implementations.

Table of Contents

Preface **7**

1. Introduction 7
2. Acknowledgements 7
3. Shell Prompts 7
4. Typographic Conventions 7
5. Notes, warnings, and examples 8

1. Introduction **1**

1.1. Copyright and Disclaimer 1
1.2. Commercial Support 1
1.3. Web Site 2
1.4. FTP Sites 2
1.5. Mailing List 2
1.6. PGP Public Key 2

2. Software Installation and Configuration **1**

2.1. Additional Recommended Software 1
2.2. Installation 2
2.3. Printer Models Supported 4

3. Recommended -Z options for Users **16**

3.1. Input Tray Selection 16
3.2. Output Bin Selection 16
3.3. Media Size (Paper) Selection 16
3.4. Media Type Selection 17
3.5. Duplex and Simplex 17
3.6. Copies 18

4. Printer Capabilities, Configurations, and Printcaps **20**

4.1. Printer Configurations 20
4.2. Network Communication Protocols 22
4.3. RFC1179 (BSD or TCP/IP) Job Transfer Printcap Entry 22
4.4. Socket Protocol (TCP/IP) Operation Printcap Entry 24
4.5. Appsocket Protocol (TCP/IP) Operation 25
4.6. Common Print Server Boxes Configuration Information 26
4.7. Timeout Problems Sending A Job 28
4.8. PS, PCL, PJL Printer with TPC/IP Network Interface 29
4.9. PS, PCL, PJL Printer with Parallel Port Connection 30
4.10. PS, PCL, PJL Printer with Serial Port 31
4.11. PostScript Only Printer 32

4.12. GhostScript....33
4.13. Tektronics Phaser, QMS, and Appsocket Protocol....34
5. Options and Arguments....35
5.1. Command Line Options....35
5.2. General Configuration Options - config, trace, debug....36
5.3. Status Messages....37
5.4. Printer Status Available - status....38
5.5. Monitoring Options - sync, waitend, pagecount....38
5.6. User -Z Option Support....39
5.7. Adding User Options....42
5.8. Initialization and Setup Control....42
6. Configuration File....44
6.1. Configuration File Entries....44
6.2. Comments....45
6.3. Option Setting....45
6.4. Option Use....46
6.5. List Expansion....46
6.6. String Escape Sequences....47
6.7. Language Context and Value Expansion....49
6.7.1. PJL Language....50
6.7.2. PCL Language....51
6.7.3. PostScript Language....51
6.8. Printer Entries....51
6.9. Include Facility....52
6.10. tc Entry Inclusion Facility....53
7. Filter Operation Details....54
7.1. Filter Pseudo-Code....54
7.2. Options, Initialization and Setup....60
7.3. Languages Supported- pjl, pcl, ps, and text....60
7.3.1. pjl_job FLAG....61
7.3.2. pjl_enter FLAG....61
7.3.3. remove_pjl_at_start FLAG....61
7.3.4. nullpad STRING....62
7.3.5. pjl_console FLAG....62
7.3.6. remove_ctrl STRING....62
7.3.7. tbcp FLAG....62
7.4. Synchronization and Pagecounts....63
7.5. PJL Initialization....67
7.6. File Conversion Support....69

7.6.1. File Type Detection 69
7.6.2. Conversion 70
7.6.3. LF to CR/LF Conversion 73
7.6.4. Text Treated Like PCL 73
7.6.5. Default to Passthrough 73
7.7. GhostScript Printer 74
7.8. Language Specific Initialization 75
7.9. File Transfer and Error Status Monitoring 76
7.10. End of Job 77
7.11. Tektronix Phaser, QMS and AppSocket Support 78
8. Banners and OF Mode Operation 80
8.1. No Banner 80
8.2. Banner Printing and No OF Filter 80
8.3. Banner Printing With OF Filter 81
8.4. **LPRng** Options Controlling Banner Printing 81
8.5. of_options option 82
9. Font Download Support 83
9.1. PCL Font Downloading 83
9.2. PS Font Downloading 84
9.3. PJL File Downloading 84
10. Debugging and Problem Solving 86
A. Index to Options 91
B. HP JetDirect Card Support 94
B.1. MicroSoft JetDirect Support 94
B.2. TCP/IP Address 94
B.3. Web Server Configuration 95
B.4. Telnet Configuration 95
B.5. BOOTP Information 95
B.6. Timeouts 100

List of Tables

2-1. Configure File Location Variables....2
2-2. Executable and Configuration File Locations....3
2-3. `ifhp.conf` Configuration Entries....5
4-1. Network Print Servers....26
A-1. `ifhp.conf` - **ifhp** Options....91

Preface

1. Introduction

The **ifhp** Print Filter is the primary print filter for the **LPRng** Print Spooler. This document is designed to be the single basic reference for the **ifhp** software; the **LPRng** software is distributed separately and has additional documentation.

2. Acknowledgements

I would like to thank all of the **LPRng** users who so relentlessly tried the incredible number of permutations and combinations of printers and software, and whose requests for *just one more feature* led to the development of the software.

3. Shell Prompts

The following table shows the default system prompt and superuser prompt. The examples will use this prompt to indicate which user you should be running the example as.

User	Prompt
Normal user	`%`
`root`	`#`

4. Typographic Conventions

The following table describes the typographic conventions used in this book.

Meaning	Examples

Meaning	Examples
The name of commands, files, and directories. On screen computer output.	Edit your `.login` file. Use `ls -a` to list all files. `You have mail.`
What you type, when contrasted with on-screen computer output.	`% su` `Password:`
Manual page references.	Use su(1) to change user names.
User and group names	Only `root` can do this.
Emphasis	You *must* do this.
Command line variables; replace with the real name or variable.	To delete a file, type `rm` *`filename`*
Environment variables	$HOME is your home directory.

5. Notes, warnings, and examples

Within the text appear notes, warnings, and examples.

Note: Notes are represented like this, and contain information that you should take note of, as it may affect what you do.

WarningWarnings are represented like this, and contain information warning you about possible damage if you do not follow the instructions. This damage may be physical, to your hardware or to you, or it may be non-physical, such as the inadvertant deletion of important files.

Examples are represented like this, and typically contain examples you should walk through, or show you what the results of a particular action should be.

Chapter 1. Introduction

The **ifhp** print filter is the latest in a long evolutionary path of print filters for the **LPRng** print spooler system. It unifies the low level printer communication facilities and provide a common code base for future development.

This document is the complete set of references and installation guide for the **ifhp** print filter. It covers compilation, installation, initial testing, details of system configuration, and configuration options that would be needed by the system administrator. Previous releases of **ifhp** had a large selection of README files which are now incorporated into the IFHP-HOWTO document.

Information about **LPRng** and **ifhp** can be found on the **LPRng** web page http://www.lprng.com.

There is mailing list for **ifhp** and **LPRng** at <`lprng@lprng.com`>. In order to reduce the amount of unsolicited *spam* mail posted to the list you must subscribe to the list before posting to it. To subscribe, send email message to *lprng-request@lprng.com (mailto:lprng-request@lprng.com)*, with the single word *subscribe* in the body.

Several presentations of **LPRng** and print spooling software have been made at the Large Scale Installation Administrator (LISA) conferences and are in the **ifhp** distribution and available on web sites. The slides for the LISA 97 tutorial on Printers and Network Print Spooling (ftp://ftp.astart.com/pub/LPRng/LISA97.tgz) are the `LISA97.ppt` in the **LPRng** distribution.

During development of **ifhp**, the following documents were invaluable references. For Printer Job Language (PJL) related issues see the Printer Job Language Technical Reference Manual, Hewlett Packard, 10th Edition, October 1997. For PCL related issues see the PCL 5 Printer Language Technical Reference Manual, First Edition, 1992. These manuals are available through the Hewlett Packard Developers Program. See http://www.hp.com/go/devexchange for information on how to join.

1.1. Copyright and Disclaimer

1.2. Commercial Support

AStArt Technologies (http://www.astart.com) (http://www.astart.com) provides commercial support and enhancements for **LPRng**, **ifhp**, and other network software. AStArt provides network and system consulting services for UNIX and NT systems, as well as real time and network software.

1.3. Web Site

Web Page: http://www.astart.com/LPRng.html

1.4. FTP Sites

Main FTP Site:

ftp://ftp.astart.com/pub/LPRng (US)

Mirrors:

ftp://ftp.cs.columbia.edu/pub/archives/pkg/LPRng (US)
ftp://ftp.cise.ufl.edu/pub/mirrors/LPRng (US)
ftp://ftp.cs.umn.edu/pub/LPRng (US)
ftp://uiarchive.uiuc.edu/pub/ftp/ftp.lprng.com/pub/LPRng (US)
ftp://ftp.sage-au.org.au/pub/printing/spooler/lprng/ (AU)
ftp://mirror.aarnet.edu.au/pub/LPRng/ (AU/NZ)
http://mirror.aarnet.edu.au/pub/LPRng/ (AU/NZ)
ftp://sunsite.ualberta.ca/pub/Mirror/LPRng (CA)
ftp://ftp.informatik.uni-hamburg.de/pub/os/unix/utils/LPRng (DE)
ftp://ftp.uni-paderborn.de/pub/unix/printer/LPRng (DE)
ftp://ftp.mono.org/pub/LPRng (UK)
ftp://ftp.iona.com/pub/plp/LPRng (IE)
ftp://uabgate.uab.ericsson.se/pub/unix/LPRng (SE)

1.5. Mailing List

To join the **LPRng** mailing list, please send mail to lprng-request@lprng.ie (mailto: lprng-request@lprng.ie) with the only the word *subscribe* in the body of the message.

1.6. PGP Public Key

The **LPRng** and **ifhp** distributions have MD5 checksum files which are signed with a PGP public key. Here is the key for validating the checksums:

```
Type Bits/KeyID    Date       User ID
pub  1024/00D95C9D 1997/01/31 Patrick A. Powell <papowell@astart.com>
   Patrick A. Powell <papowell@sdsu.edu>

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i

mQCNAzLygTQAAAEEANBW5fPYjN3wSAnP9xWOUc3CvsMUxjip0cN2sY5qrdoJyIhn
qbAspBopR+tGQfyp5T7C21yfWRRnfXmoJ3FVtgToAsJUYmzoSFY08eDx+rmSqCLe
rdJjX8aG8jVXpGipEo9U4QsUK+OKzx3/y/OaK4cizoWqKvy1l4lEzDsA2VydAAUT
tCdQYXRyaWNrIEEuIFBvd2VsbCA8cGFwb3dlbGxAYXN0YXJ0LmNvbT6JAJUDBRA0
XonoiUTMOwDZXJ0BAQ2cBAC7zU9Fn3sC3x0USJ+3vjhg/qA+Gjb5Fi1dJd4solc4
vJvtf0UL/1/rGipbR+A0XHpHzJUMP9ZfJzKZjaK/d0ZBNlS3i+JnypypeQiAqo9t
FV0OyUCwDfWybgAORuAa2V6UJnAhvj/7TpxMmCApolaIb4yFyKunHa8aBxN+17Ro
rrQlUGF0cmljayBBLiBQb3dlbGwgPHBhcG93ZWxsQHNkc3UuZWR1PokAlQMFEDLy
gTSJRMw7ANlcnQEBYBYD/0zTeoiDNnI+NjaIei6+6z6oakqO70qFVx0FG3aP3kRH
WlDhdtFaAuaMRh+RItHfFfcHhw5K7jiJdgKiTgGfj5Vt3OdHYkeeh/sddqgf9YnS
tpj0u5NfrotPTUw39n6YTgS5/aW0PQfO9dx7jVUcGeod1TGXTe9mIhDMwDJI4J14
=3Zbp
-----END PGP PUBLIC KEY BLOCK-----
```

Chapter 2. Software Installation and Configuration

Before you do an installation please read the following instructions. You will need to:

1. Use GNU Make. You can get it from http://www.gnu.org/software/software.html. Don't even think about trying to use another make unless you are a Wizard. And even the Wizards use GNU Make.
2. Use an ANSI C compiler. **ifhp** is developed and tested with the GNU C compiler. You can get it from http://www.gnu.org/software/software.html. Solaris users should consult the excellent http://sunfreeware.com site for binary distributions. AIX users can use http://www.bull.de/pub/ or ftp://ftp.htwk-leipzig.de

2.1. Additional Recommended Software

The following software is recommended for use with **ifhp**. If your printer does not support PostScript, PCL, or text printing directly you will need to install GhostScript to convert from PostScript to the printer format and use a suitable text to PostScript converter.

Unix File Utility

The **ifhp** filter will recognize that a file is PostScript, PJL, or PCL by examining the first couple of bytes of a file and applying a simple set of rules. If you require more elaborate file type detection then you can configure **ifhp** to also use the UNIX *file* utility if it is unable to determine the file type. See ftp://ftp.astron.com/pub/file/ or ftp://ftp.lprng.com/pub/LPRng/UNIXTOOLS/file/ to obtain this software.

LPRng Print Spooler

http://www.astart.com/LPRng.html The **ifhp** filter works best with the later versions of this software, and the two are developed as an integrated unit.

GhostScript

http://www.cs.wisc.edu/~ghost/index.html or http://www.ghostscript.com (http://www.ghostscript.com/) If your printer does not handle PostScript and you need to print PostScript, GhostScript is used to convert PostScript to a format usable by the printer.

a2ps - Ascii Text To PostScript Converter

If your printer is a PostScript only printer or you wish to have enhanced formatting capability for documents, then you will need a text to PostScript converter. http://www-inf.enst.fr/~demaille/a2ps/ This package does a very good job of text to PostScript conversion. It also makes use of the file utility to determine the required conversions.

enscript - GNU Enscript

http://www.gnu.org/ (http://www.gnu.org) This package is an alternative to `a2ps`, but requires careful handling due to the exit codes it produces. Please see Wrappers For Programs for details on how to use `enscript` with **ifhp**.

textps

This program is included with the **ifhp** distribution and is an extremely primitive text to PostScript filter. It generates PostScript compatible with even the most ancient of PostScript printers and is useful where `a2ps` and *enscript* are just too modern. See http://www.astart.com/LPRng.html.

psutils

The `psutils` package developed by Angus Duggan is available from ftp://ftp.dcs.ed.ac.uk/pub/ajcd/. These are a collection of programs for manipulation of PostScript files, and include facilities for doing page selection, page reversal, n-up printing, and watermarking.

netcat

The `netcat` utility is extremely useful when trying to send files to a network printer and you need to monitor its activity. Developed by <`hobbit@avian.org`>, it is available from ftp://avian.org/src/hacks/nc110.tgz.

2.2. Installation

The installation procedure uses the `configure` facility to generate Makefiles. By convention, these files have the following variables that install the **ifhp** executables and configuration files in the following locations:

Table 2-1. Configure File Location Variables

Configure Variable	Default Value	Expanded Default Value	Override

Configure Variable	Default Value	Expanded Default Value	Override
${prefix}	/usr/local	--prefix=PATH	
${exec_prefix}	${prefix}	/usr/local	--execprefix=PATH
${bindir}	${exec_prefix}/bin	/usr/local/bin	--bindir=PATH
${sbindir}	${exec_prefix}/sbin	/usr/local/sbin	--sbindir=PATH
${libexecdir}	${exec_prefix}/libexec	/usr/local/libexec	--libexecdir=PATH
${sysconfdir}	${prefix}/etc	/usr/local/etc	--sysconfdir=PATH
${mandir}	${prefix}/man	/usr/local/man	--mandir=PATH

The following files are installed as shown below:

Table 2-2. Executable and Configuration File Locations

Configure Variable	Files
${libexecdir}/filters	lpf, ifhp
${sysconfdir}	ifhp.conf
${mandir}/man[1-9]	man pages

The configuration you choose should match that of the **LPRng** print spooler. For example:

```
./configure --prefix=/usr --sysconfdir=/etc \
  --mandir=/usr/share/man

executables and files in
   /usr/libexec/filters         ifhp
   /etc                         ifhp.conf
   /usr/share/man/man[0-9]      ifhp.man1
```

First, we untar, configure, compile, and install the software:

```
h4: {1} % gunzip -c ifhp-<version>.tgz | tar xvf -
h4: {2} % cd ifhp-<version>
h4: {3} % ./configure  [ ... configuration options ]
h4: {4} % make clean all
h4: {5} % su   # you must do the following commands as root
```

```
h4: {6} # make install
```

Modify your printcap file to use **ifhp**. Your printcap usually has the following format; older version of **lpd** require *:* at the end of each line of a printcap entry. The *:if* and *:of* filter entries are usually the ones of interest.

```
lp:
  :lp=xxxx:sd=xxxx:....
  :if=/usr/local/path_to_old_filters/old_if_filter
  :of=/usr/local/path_to_old_filters/old_of_filter
```

Your new printcap entry will look like the one below. The MODEL information is described in the next section.

```
lp:
  :lp=xxxx:sd=xxxx:....
  #  see text for details about the next line
  :ifhp=model=MODEL,status@
  :filter=/usr/local/libexec/filters/ifhp
  # only if you are using accounting or banners
  #:of=/usr/local/libexec/filters/ifhp
```

Select a suitable printer from the entries in the configuration file (`/usr/local/etc/lpd.conf` or `/etc/lpd.conf`). These are described in more detail in the next section.

Shut down and restart your print spooler and then send a job to the printer. If this works and you do not need any further capabilities of **ifhp** such as error reporting or printer monitoring, then you are finished.

If you want to use additional capabilities, then you should read the detailed instructions in the next couple of sections.

2.3. Printer Models Supported

There are over 500 different printer models, types and configurations supported by **ifhp**. If your printer is not currently supported and you have documentation about the printer then send mail to the **LPRng** Mailing List and support will be added.

The default printer is an HP LaserJet 4MP, which supports PostScript Level 3, PCL5, PJL, and has bidirectional communication and a functional pagecounter.

The `ifhp.conf` configuration file contains configuration entries for various models of printers. Each entry has a name usually corresponding to the model of printer or its basic capabilities. For example, the HP LaserJet 4 printer has the `model=hp4` configuration entry. The *default* printer configuration covers a wide range of network printers manufactured by Hewlett-Packard, Canon, Epson, and others and is for a printer that has a bidirectional communications connection that allows it to report status information and the following capabilities:

1. PJL support (`pjl`) compatible with HP 4 family of printers
2. PostScript (PS) support (`ps`).
3. PCL support (`pcl`).
4. Text files printed as PCL (`text`, `default_language=pcl`).

There is also support for PostScript only printers (`model=ps`), Tektronics Phasers (`model=phaser`), QMS (`model=qmsXXX`) and others. The best way to determine the printers currently supported is to examine the `ifhp.conf` file. The following is a sample of the various entries in the configuration file.

Table 2-3. `ifhp.conf` Configuration Entries

Configuration	Printer Supported
`default`	HP 4M Plus, PostScript, PJL, PCL, status, pagecount support
`apple`	PostScript printer, text to PS conversion, status, pagecount support
`postscript`	PostScript printer, text to PS conversion, status, pagecount support
`ps`	PostScript printer, text to PS conversion, status, pagecount support
`pcl`	PCL only printer, no status
`pcl_gs`	HP Laserjet 4 PCL only printer, write only, no status
`hpiiisi`	HP LaserJet III (PCL and PostScript Interpreter)
`hp3si`	HP LaserJet III (PCL and PostScript Interpreter)
`ljet3`	HP LaserJet III (PCL and PostScript Interpreter)
`lj3`	HP LaserJet III (PCL and PostScript Interpreter)

Configuration	Printer Supported
`hpiiisi.gs`	HP LaserJet III (PCL, PostScript via GhostScript)
`hp3si.gs`	HP LaserJet III (PCL, PostScript via GhostScript)
`ljet3.gs`	HP LaserJet III (PCL, PostScript via GhostScript)
`lj3.gs`	HP LaserJet III (PCL, PostScript via GhostScript)
`hp4`	HP LaserJet 4 Family, HP LaserJet 4 (PostScript Interpreter)
`hp4m`	HP LaserJet 4 Family, HP LaserJet 4m (PostScript Interpreter)
`hp4si`	HP LaserJet 4 Family, HP LaserJet 4si (PostScript Interpreter)
`hp4simx`	HP LaserJet 4 Family, HP LaserJet 4simx (PostScript Interpreter)
`hp4plus`	HP LaserJet 4 Family, HP LaserJet 4plus (PostScript Interpreter)
`hp4mplus`	HP LaserJet 4 Family, HP LaserJet 4mplus (PostScript Interpreter)
`hp4v`	HP LaserJet 4 Family, HP LaserJet 4v (PostScript Interpreter)
`hp4mv`	HP LaserJet 4 Family, HP LaserJet 4mv (PostScript Interpreter)
`hp4p`	HP LaserJet 4 Family, HP LaserJet 4p (PostScript Interpreter)
`hp4mp`	HP LaserJet 4 Family, HP LaserJet 4mp (PostScript Interpreter)
`hp4pj`	HP LaserJet 4 Family, HP LaserJet 4pj (PostScript Interpreter)
`hpljpro`	HP LaserJet 4 Family, HP LaserJet ljpro (PostScript Interpreter)
`hp4lc`	HP LaserJet 4 Family, HP LaserJet 4lc (PostScript Interpreter)
`hp4mplus`	HP LaserJet 4MPlus
`hp5l`	HP LaserJet 5 Family, Hp Laserjet 5l
`hp6l`	HP LaserJet 6 Family, Hp Laserjet 6l
`hp1100`	HP LaserJet 6 Family, Hp Laserjet 1100

Configuration	Printer Supported
hp1100	HP LaserJet 1000 Family, Hp Laserjet 1100
hp4l	Hp LaserJet 4L, PCL only
hp4ml	HP LaserJet 4ml
hp5p	HP LaserJet 5p
hp5mp	HP LaserJet 5mp
hp6p	HP LaserJet 6p
hp6mp	HP LaserJet 6mp
hp5	HP LaserJet 5
hp5si	HP LaserJet 5si
hp5simx	HP LaserJet 5simx
hp5m	HP LaserJet 5m
hp5simopier	HP LaserJet 5simopier
hp4000	HP LaserJet 4000
hpcolorlj	HP Color LaserJet
hpcolorlj5	HP Color LaserJet 5
hpcolorlj5m	HP Color LaserJet 5m
hpdj1200	HP Design Jet 1200 Family
hppjxl300	HP Paint Jet XL 300 Family
hpdj1600	HP Design Jet 1600 Family
hpdj200	HP DeskJet 200
hpdj220	HP DeskJet 220
hpdj600	HP DeskJet 600
hpdj650	HP DeskJet 650
hpdj230	HP DeskJet 230
hpdj250c	HP DeskJet 250c
hpdj330	HP DeskJet 330
hpdj350c	HP DeskJet 350c
hpdj430	HP DeskJet 430
hpdj450c	HP DeskJet 450c
hpdj455ca	HP DeskJet 455ca
hpdj700	HP DeskJet 700
hpdj750c	HP DeskJet 750c

Configuration	Printer Supported
`hpdj750cplus`	HP DeskJet 750cplus
`hpdj755cm`	HP DeskJet 755cm
`hpdj2000cp`	HP DeskJet 2000cp
`hp2500`	HP Design Jet 2500 - No PCL, PostScript Only
`hp2500cm`	HP Design Jet 2500cm - No PCL, PostScript Only
`hp2500c`	HP Design Jet 2500c - No PCL, PostScript Only
`hp2500cm`	HP Design Jet 2500cm - PCL and PostScript
`hpdj2500cp`	HP DesignJet 2500CP, not HP2500, HP2500c, HP2500cm
`hp4500`	HP Color LaserJet Printer 4500
`hp8500`	HP Color LaserJet Printer 8500
`hp8550`	HP Color LaserJet Printer 8550
`hp5000`	HP5000 Model number: C4111A (LaserJet 5000N)
`hp5000`	HP5000 Model C4111A (LaserJet 5000)
`hp8000`	HP Laserjet 8000 Series, HP8000
`hp8100`	HP Laserjet 8000 Series, HP8100
`hp8150`	HP Laserjet 8000 Series, HP8150
`hp2100`	HP LaserJet 2100 Series
`hp2200`	HP LaserJet 2200 Series
`hp4050`	HP4050 Series
`hp4050`	HP4050 Series Printers
`qms1725`	QMS 1725, uses appsocket, no status, PostScript only
`qms2025`	QMS Laser Printer QMS 2025, appsocket, no status, PostScript and PCL
`qms860`	QMS Laser Printer QMS 860, appsocket, no status, PostScript and PCL
`qms2060`	QMS Laser Printer QMS 2060, appsocket, no status, PostScript and PCL
`phaser`	Generic Tektronics Phaser Color Printer, appsocket, PostScript only
`phaser360`	Tektronics Phaser 360 Color Printer, appsocket, PostScript only

Configuration	Printer Supported
`phaser740`	Tektronics Phaser 740 Color Printer, appsocket, PostScript only
`phaser750`	Tektronics Phaser 750 Color Printer, appsocket, PostScript only
`phaser850`	Tektronics Phaser 850 Color Printer, appsocket, PostScript only
`lexmark4039`	Lexmark 4039, Postscript only
`lexmark_optra_e312`	Lexmark Optra e312, Postscript, PCL and PJL
`ln15s`	Digital Laser LN15, LN17ps, Compaq Laser LN 16
`ln16s`	Digital Laser LN15, LN17ps, Compaq Laser LN 16
`ln17pss`	Digital Laser LN15, LN17ps, Compaq Laser LN 16
`hpij2250s`	HP Business Inkjet 2250
`gs_bj10`	Canon BubbleJet BJ10e
`gs_bj10`	Canon BubbleJet BJ20
`gs_bj200`	Canon BubbleJet BJ200
`gs_bj200`	Canon BubbleJet BubbleJet BJC-210 B/W only
`gs_bj200`	Canon BubbleJet BubbleJet BJC-240 B/W only
`gs_bj200`	Canon BubbleJet BubbleJet BJC-250 B/W only
`gs_bj200`	Canon BubbleJet BubbleJet BJC-70 B/W only
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-600
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-610
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-50
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-70
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-80
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-210 Color only
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-240 Color only
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-250
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-1000
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-2000
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-4000
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-4100 B/W only
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-4200
`gs_bjc600`	Canon BubbleJet BubbleJet BJC-4300

Configuration	Printer Supported
gs_bjc600	Canon BubbleJet BubbleJet BJC-4550
gs_bjc600	Canon BubbleJet BubbleJet BJC-6000
gs_bjc600	Canon MultiPASS C2500 color printer/fax/copier
gs_bjc800	Canon BubbleJet BubbleJet BJC-800
gs_bjc800	Canon BubbleJet BubbleJet BJC-7000 Color.
gs_bjc800	Canon BubbleJet BubbleJet BJC-4300 Color
gs_bjc800	Canon BubbleJet BubbleJet BJC-4650
gs_deskjet	HP DeskJet
gs_deskjet	HP DeskJet Plus
gs_djet500	HP DeskJet 500 B/W
gs_djet500	HP DeskJet Portable B/W
gs_djet500	HP OfficeJet 590 B/W
gs_cdj500	HP DeskJet 400
gs_cdj500	HP DeskJet 500C
gs_cdj500	HP DeskJet 540C
gs_cdj500	HP DeskJet 690C
gs_cdj500	HP DeskJet 693C
gs_cdj550	HP DeskJet 550C
gs_cdj550	HP DeskJet 560C
gs_cdj550	HP DeskJet 600
gs_cdj550	HP DeskJet 660C
gs_cdj550	HP DeskJet 660C
gs_cdj550	HP DeskJet 682C
gs_cdj550	HP DeskJet 683C
gs_cdj550	HP DeskJet 693C
gs_cdj550	HP DeskJet 694C
gs_cdj550	HP DeskJet 690C
gs_cdj550	HP DeskJet 692C
gs_cdj550	HP DeskJet 693C
gs_cdj550	HP DeskJet 694C
gs_cdj550	HP DeskJet 695C
gs_cdj550	HP DeskJet 850

Configuration	Printer Supported
gs_cdj550	HP DeskJet 870Cse
gs_cdj550	HP DeskJet 895Cxi
gs_cdj550	HP DeskJet 970
gs_cdj550	HP OfficeJet 590
gs_cdj550	Olivetti jp450
gs_cdj550	Xerox XJ6C
gs_cdj850	HP DeskJet 850
gs_cdj850	HP DeskJet 855
gs_cdj850	HP DeskJet 870Cse
gs_cdj850	HP DeskJet 870Cxi
gs_cdj850	HP DeskJet 890C
gs_cdj850	HP DeskJet 670C
gs_cdj850	HP DeskJet 680
gs_cdeskjet	HP DeskJet 500C
gs_cdeskjet	GhostScript with -sDEVICE=cdj500 -dBitsPerPixel=3
gs_cdjcolor	GhostScript with -sDEVICE=cdj500 -dBitsPerPixel=24
gs_cdjmono	HP DeskJet 500C
gs_cdjmono	HP DeskJet 510
gs_cdjmono	HP DeskJet 520
gs_cdjmono	HP DeskJet 540C
gs_cdjmono	HP DeskJet 693C
gs_cdjmono	GhostScript with -sDEVICE=cdj500 -dBitsPerPixel=1
gs_epsonc	Fujitsu DL-1100
gs_epsonc	Fujitsu DL-2400
gs_hl7x0	Brother HL-720
gs_hl7x0	Brother HL-730
gs_hl7x0	Do not use hl7x0 with PCL compliant Brother HL-760. Use ljet4.
gs_laserjet	Bull Compuprint Pagemaster 415
gs_lips3	Canon LBP4+

Configuration	Printer Supported
gs_lj4dith	HP DeskJet 600
gs_ljet2	HP LaserJet II
gs_ljet2	Xerox 4030
gs_ljet3	Tandy LP800 With LaserJet III emulation.
gs_ljet4	Brother HL-660
gs_ljet4	Brother HL-760 600dpi
gs_ljet4	Epson EPL5700 300dpi OK
gs_ljet4	HP DeskJet 600 margins wrong
gs_ljet4	HP DeskJet 870Cse
gs_ljet4	HP LaserJet 5 300dpi or 600dpi
gs_ljet4	HP LaserJet 5L 300dpi or 600dpi
gs_ljet4	HP LaserJet 6L 600dpi
gs_ljet4	HP LaserJet 1100 600dpi OK.
gs_ljet4	IBM Network Printer 17
gs_ljet4	IBM/Lexmark 4029 Margins wrong.
gs_ljet4	Lexmark Optra E+
gs_ljet4	Lexmark Optra SC 1275 B/W only.
gs_ljet4	Oki OL410ex LED printer 300dpi or 600dpi
gs_ljetplus	HP LaserJet Plus
gs_ljetplus	Canon Laser LBP-600
gs_ljetplus	NEC SuperScript 860
gs_pjxl300	HP PaintJet XL300
gs_pjxl300	HP DeskJet 600
gs_pjxl300	HP DeskJet 1200C
gs_pjxl300	HP DeskJet 1600C
gs_r4081	Ricoh 4081 laser printer
gs_r4081	Ricoh 6000 laser printer
bjc610a0.upp	Canon BubbleJet BJC 610 (color, rendered) 360x360dpi plain paper, high speed
bjc610a1.upp	Canon BubbleJet BJC 610 (color, rendered) 360x360dpi plain paper
bjc610a2.upp	Canon BubbleJet BJC 610 (color, rendered) 360x360dpi coated paper

Configuration	Printer Supported
`bjc610a3.upp`	Canon BubbleJet BJC 610 (color, rendered) 360x360dpi transparency film
`bjc610a4.upp`	Canon BubbleJet BJC 610 (color, rendered) 360x360dpi back print film
`bjc610a5.upp`	Canon BubbleJet BJC 610 (color, rendered) 360x360dpi fabric sheet
`bjc610a6.upp`	Canon BubbleJet BJC 610 (color, rendered) 360x360dpi glossy paper
`bjc610a7.upp`	Canon BubbleJet BJC 610 (color, rendered) 360x360dpi high gloss film
`bjc610a8.upp`	Canon BubbleJet BJC 610 (color, rendered) 360x360dpi high resolution paper
`bjc610b1.upp`	Canon BubbleJet BJC 610 (color, rendered) 720x720dpi plain paper
`bjc610b2.upp`	Canon BubbleJet BJC 610 (color, rendered) 720x720dpi coated paper
`bjc610b3.upp`	Canon BubbleJet BJC 610 (color, rendered) 720x720dpi transparency film
`bjc610b4.upp`	Canon BubbleJet BJC 610 (color, rendered) 720x720dpi back print film
`bjc610b6.upp`	Canon BubbleJet BJC 610 (color, rendered) 720x720dpi glossy paper
`bjc610b7.upp`	Canon BubbleJet BJC 610 (color, rendered) 720x720dpi high-gloss paper
`bjc610b8.upp`	Canon BubbleJet BJC 610 (color, rendered) 720x720dpi high resolution paper
`cdj550.upp`	HP DeskJet 550C 300x300dpi 32-bit CMYK
`necp2x.upp`	NEC P2X 360x360dpi 8-bit (Floyd-Steinberg)
`stcany.upp`	Epson Stylus Color (Any) 360x360dpi 4-bit, PostScript halftoning
`stc.upp`	Epson Stylus (Original) and Stylus Pro Color 360x360dpi 32-bit CMYK, 15-pin
`stc_l.upp`	Epson Stylus (Original) and Stylus Pro Color 360x360dpi 4-bit, PostScript halftoning, weaved noWeave

Configuration	Printer Supported
`stc_h.upp`	Epson Stylus (Original) and Stylus Pro Color 720x720dpi 32-bit CMYK, 15-pin Weave
`stc2.upp`	Epson Stylus (Original) and Stylus Pro Color 360x360dpi 32-bit CMYK, 20-pin, Epson Stylus Color II(s)
`stc2_h.upp`	Epson Stylus (Original) and Stylus Pro Color 720x720dpi 32-bit CMYK, 20-pin, Epson Stylus Color II
`stc2s_h.upp`	Epson Stylus (Original) and Stylus Pro Color 720x720dpi 32-bit CMYK, 20-pin, Epson Stylus Color IIs
`stc500p.upp`	Epson Stylus Color 500 360x360dpi 32-bit CMYK, noWeave, plain paper
`stc500ph.upp`	Epson Stylus Color 500 720x720dpi 32-bit CMYK, noWeave, plain paper
`stc600pl.upp`	Epson Stylus Color 600, 360x360dpi, 32/90-inch weaving 32-bit CMYK, 32-pin, plain paper
`stc600p.upp`	Epson Stylus Color 600, 720x720dpi, 32/90-inch weaving 32-bit CMYK, 32-pin, plain paper
`stc600ih.upp`	Epson Stylus Color 600, 1440x720dpi, 32/90-inch weaving 32-bit CMYK, 30-pin, inkjet paper
`stc800pl.upp`	Epson Stylus Color 800, 64/180-inch weaving 360x360dpi 32-bit CMYK, 64-pin, plain paper
`stc800p.upp`	Epson Stylus Color 800, 64/180-inch weaving 720x720dpi 32-bit CMYK, 64-pin, plain paper
`stc800ih.upp`	Epson Stylus Color 800, 64/180-inch weaving 1440x720dpi 32-bit CMYK, 62-pin, inkjet paper
`stc1520h.upp`	Epson Stylus Color 800, 64/180-inch weaving 1440x720dpi 32-bit CMYK, 62-pin, inkjet paper

If your printer is not in this this list then you can use the following guidelines. If you have a PostScript only printer you should use the `ps` model. If you have a PCL only printer, then `pcl` is recommended. If you want to process PostScript files on your PCL only printer then install GhostScript and use pcl_ps entry and select the GhostScript driver suitable for your printer.

The other model entries are used when specific printer functionality or features are needed. For example,

if you want to do accounting or use *landscape* mode, then you should check for your specific printer model in the configuration file.

Chapter 3. Recommended -Z options for Users

Due to the general nature of the **ifhp** filter, there is no standard set of `-Z` user options because there is no standard set of user facilities. However, the following are recommended for use by implementors of new configurations or printer support.

3.1. Input Tray Selection

If a printer supports an input tray selection mechanism, then the following options are recommended for use. Local conditions or printer type may require addition options.

`inupper, inlower, intray1, intray2, ..., manual, envelope`

The input tray selection options should start with the `in` prefix and correspond to the various trays, if possible. The `manual` and `envelope` options are included to select manual feed or envelope feed. There is a possible source of conflict here as there may be an envelope feeder as well as an envelope media. This is a printer specific dependency.

`source=name`

The `key=value` form allows users to use options such as `-Zsource=inbin1`, which may be useful for systems that have an unusual or nonstandard input selection mechanism.

3.2. Output Bin Selection

If a printer has an output bin selection mechanism or some other finishing mechanism, then the following are recommended for use.

`outupper, outlower, outbin1, ...`

The output bin selection should start with the `out` prefix.

`outbin=name`

The `outbin=name` form allow users to use options such as `-Zoutbin=stapler`, which may be useful for systems that have an unusual or nonstandard output selection mechanism.

3.3. Media Size (Paper) Selection

The paper size selection facilities usually are quite printer dependent, and the input tray selection and paper size selection mechanisms may interact in strange and mysterious ways.

`letter, legal, ledger, oversize, a0, a1, ...`

These are standard paper size names.

`11x17, tabloid`

These are usually aliases for ledger, but depending on local conditions can select different types of paper.

`paper=name`

The `paper=name` form allow users to use options such as `-Zpaper=b3`, which may be useful for systems that have an unusual or nonstandard input media selection mechanism.

3.4. Media Type Selection

Media Type is not the same as paper size, and corresponds to the name assigned to a particular media. Of course, the issue is complicated by the fact that some media have standard sizes as well. Again, the input tray selection, media size, and media type selection will interact in confusing and mysterious ways, depending on the whim of the printer firmware implementors.

You will also notice that there is no general `mediatype=name` selection mechanism. This is due to the extremely different way that the media names must be passed for PostScript, PJL, and PCL.

`plain, preprinted, letterhead, transparency, glossy, prepunched, labels`

These are commonly used media type names gleaned from various PostScript Printer Description Files, Microsoft printer drivers, and arcane lore of the Printer Working Group. Note that these are not accepted terms in the paper industry for any of these type of media. You are warned.

3.5. Duplex and Simplex

Duplex printing is when impressions are placed on both sides of a sheet of media. Due to a general lack of conventions, the orientation of each of the impressions varies from vendor to vendor, and has changed over the years.

`duplex, lduplex`

Print on both sides using the default orientation. The `lduplex` is an alias for `duplex`

`duplexshort, sduplex`

Print on both sides but reverse the orientation of one page. The `sduplex` is an alias for `duplexshort` Which page is reversed is at the whim of the firmware implementors and conventions for the printer.

`simplex`

Print on a single side of a page

`tumble, shortedge`

This is use to print a single page on one side of the media, but using the (nonstandard) orientation for the `duplexshort`. This is usually done when a single impression must be generated on the alternative side of the media, rather than the default side. Again, this is dependent on the whims and whimsies of the printer firmware implementors, and may have some unexpected side effects.

3.6. Copies

This option has been provided to effectively allow the printer to make multiple copies of a single page or job. This option tends to be misimplemented on almost all known printers, and it is strongly recommended that users do not use it. However, for completeness, compatibility, and implementor consideration, this is included, even against the better judgement of the implementors of the **ifhp** software.

`copies=nnn`

Attempt to make `nnn` copies of each impression. This usually fails with catastrophic problems unless you have a system that supports all of the various options required, has enough memory to

handle rasterization, you do not have a paper outage, and the printer does not stop with operator intervention. You have been warned.

Chapter 4. Printer Capabilities, Configurations, and Printcaps

One of the major difficulties with printer software is dealing with the wide range of different printer hardware configurations and printer connections. This section outlines the printer communication methods, the types of print job languages, and the effects of these on printing software and the **ifhp** filter.

4.1. Printer Configurations

A printer consists of a hardware print engine which marks the output page and delivers it, a set of control hardware that takes a *print job* in a well defined format and operates the hardware to produce output according to information in the *print job*, and a communication channel from the computer to the control hardware. The control hardware is sometimes called a *print engine*. In most modern computers the control hardware may consist of multiple microprocessors, each with their own firmware, and each performing a specific printing task. For example, one may control the paper feed path, one may do rasterization, and one may handle communications with the outside world.

In order to set up printing correctly, it is necessary to know the following information about your printer.

1. The capabilities of the hardware. This is dependent on the model of printer, and may be such things as the page feed, output and input tray selection, numbers of columns and/or rows of output available on the output device. This information is readily available from most manufacturers.
2. The *print job language* recognized by the control hardware. This is the special set of codes, commands, and formats recognized by the control hardware.
3. The protocol used to send jobs to the printer and obtain status about the printing activity.

Usually the capabilities of a modern printer are very well known and documented, and the **ifhp** filter and most print spooling software has little difficulty working with them.

The following checklist will help you in setting up your printer. The various options that you will need to know about are indicated where appropriate.

1. Printer Model (`model=???`) What is the exact printer model? Check the serial number or other identification to get this information. You should check the `ifhp.conf` configuration file to see if your printer is already supported.
2. Print Languages Supported By Your Printer

a. PJL? (`pjl` or `pjl@`) The Printer Job Language (PJL) is a high level language supported by many Hewlett-Packard printers that allows some print system configuration to be performed. Due to historical developments, not all printers support all PJL language facilities, and some support them in different ways than other printers. The **ifhp** filter can use the PJL support for a printer if it is available.

b. PostScript (and what version)? (`ps` or `ps@`) PostScript is the most common print job language in use. If your printer supports PostScript, then you will have a relatively trouble free time with it. One problem is that it requires a fairly substantial amount of memory and computational support, and is usually not found on the low end (less than $500) printers.

c. PCL? (`pcl` or `pcl@`) PCL is another Print Language supported by many vendors, including Hewlett-Packard, Lexmark, and others. It is essentially text with escape sequences to tell the print engine to place markings on a page at specific places in a specific font. It is the second most common format used with modern printers.

d. Text? (`text` or `text@`) Text is really just PCL without any control sequences. However, it is easy to have **ifhp** convert ordinary text into PCL by prefixing the appropriate PCL control codes. You may also need to use the `crlf` option to force CR to `CR-LF` translation. If you have a simple text printer then you may want to use the much easier to configure **lpf** filter from the **LPRng** distribution (http://www.astart.com/LPRng.html).

e. Vendor Specific There is a growing trend to have very proprietary Print Languages for very low end (less than $300) printers. These printers usually require all of their jobs to be preformated by software running on the host and to have the job delivered to them in a specific manner. If you have one of these printers, you will need to get a rasterizing program that produces the correct format. Check to see if GhostScript, supports your printer. If it does then you can use `GhostScript` to translate PostScript to your printer's required format.

3. Memory Size. If you are going to be sending large print jobs or ones with a large amount of graphics to the printer, you will need a substantial amount of memory to deal with rasterization. Most high resolution Laser Copier based printers require a minimum of 16 megabytes for adequate performance, and if you are printing complex PostScript or PDF documents you may want at least 32 megabytes. Color printers require substantially more and 64 megabytes is not uncommon.

4. Communications. The connection between your printer and the host computer.

   a. Network Connection This is the most reliable and high speed way to connect a printer to a system. This is especially true if a printer must be accessible to multiple users and is located at a distance from the user.

   b. Parallel Port (`status@`) The parallel port is a *unidirectional* communications channel and does not do full duplex bidirectional communications. Some operating system support bidirectional communications, but they do so by requiring write operations to alternate with read operations.

c. Serial Ports This is the very worst way to communicate at high speed with a printer. Serial ports usually have a high error rate, suffer from data overruns, and have a severe impact on system performance. You will need to configure your printer speed, format (bits per character, parity, stop bit), and flow control method, and then do the same for the host. This can be an endless source of frustration for the novice user.

d. Print Server Box Many older printers do not directly support a network connection and have an external *print server box* attached to either their serial or parallel ports. If you have the printer connected to a parallel port, then you will still most likely only have unidirectional communication and no status information will be available from the printer.

4.2. Network Communication Protocols

The most high speed and reliable connection to your printer is using a network connection. The following protocols are usually used to communicate with a network printer: RFC1179 (TCP/IP printing), Socket Protocol (TCP/IP), AppSocket Protocol (TCP/IP), Novell Print Protocol (IPX), SMB Print Protocol (TCP/IP), and AppleTalk Print Protocol (TCP/IP).

It is highly recommended that you use TCP/IP networking to communications to talk to your printer, and that you do not enable any other protocol on your printer. If you have two different systems trying to connect to the same printer using different protocols, a wide range of vendor's hardware will lock up and may require a power up reset to recover. Documented evidence for this behavior includes a wide range of printers, including those from Hewlett-Packard, LexMark, IBM and other vendors.

Only the TCP/IP based network job transfer protocols are discussed in this document. For details on using other protocols, please consult the consult the **LPRng** (http://www.astart.com/LPRng.html) documentation.

4.3. RFC1179 (BSD or TCP/IP) Job Transfer Printcap Entry

RFC1179 is used to transfer print jobs between a client (user) and a print spooler, or between two print spoolers. Jobs are transferred as a set of files, and the only information exchanged during the transfer process is the success or failure of the transfer. In order to get status about the actual job printing, a separate query status (**lpq**) is sent to the print spooler.

Many, if not all, printers with a network interface that supports the TCP/IP protocol support the RFC1179 protocol for job transfer. However, their support for print job status is usually minimal to non-existent. If you want to send a job to a printer using the RFC1179 protocol, please be aware of the following problems.

Normally a print spooler (System 5 lp, BSD lpd, **LPRng**) does not modify a print job when forwarding it to another print spooler. This means that your print job will normally pass from the originating **lp** or **lpr** program to the destination printer with no changes. This can have disastrous results if the job *requires* filter processing.

If you are using the **LPRng** print spooler, job transfers using `RFC1179` is specified by using `:lp=spoolqueue@host` or `:rp=spoolqueue:rh=host` printcap entries. For example:

```
raw:
  :lp=raw@host
  :sh:sf:mx=0
  :sd=/var/spool/lp
cooked:
  :rp=cooked:rm=host
  :sh:sf:mx=0
  :sd=/var/spool/lp
```

No filters are specified as the job is not modified, only transferred from one server to another. Even if filters were specified they would be ignored. The lpd_bounce flag causes the **LPRng** spooler to pass the print job through the specified filter and then send the filter output to the actual network printer. The **lpd** print spooler will open a temporary file to hold the filter output, and then proceed to start the specified filter with its STDOUT attached to the temporary file, its STDIN attached to the file to be processed, and its STDERR redirected to an error log. The single resulting file is then transferred to the destination system using the `RFC1179` protocol.

When a job is created the job format is specified (default is `f`), and a filter named by the `:i`*<format>* option is selected for use. For example:

```
raw:
  :lpd_bounce
  :lp=raw@host
  :sh:sf:mx=0
  :sd=/var/spool/lp
  :ifhp=model=XXX,status@
  # for 'f' format
  :filter=/usr/local/libexec/filters/ifhp
cooked:
  :lpd_bounce
```

```
  :rp=cooked:rm=host
  :sh:sf:mx=0
  :sd=/var/spool/lp
  :ifhp=model=XXX,status@
  # for 'f' format
  :filter=/usr/local/libexec/filters/ifhp
```

Unfortunately, some print spooling systems also use the `v` format by default. You may find the following printcap entry useful in this case. The `:filter` option specifies a default filter that is used if one is not specified for the format.

```
raw:
  :lpd_bounce
  :lp=raw@host
  :sh:sf:mx=0
  :sd=/var/spool/lp
  :ifhp=model=XXX,status@
  # for 'f' format
  :filter=/usr/local/libexec/filters/ifhp
```

The `lpr -l` or `lpr -b` flag is used to specify that a job has the special `binary` flag. In this case, most filters will perform only the most perfunctory processing and pass the job directly to the printer.

4.4. Socket Protocol (TCP/IP) Operation Printcap Entry

Many printers with a network interface provide a TCP/IP port that is a direct connection to the internal *print engine*. If a TCP/IP connection is made to this port and a file is sent over this connection, then the print engine will process the file. More importantly, the connection is bidirectional, and the printer will report errors and status conditions over the connection. PJL and PostScript status request commands can be sent to the printer and the printer will respond with information.

The **ifhp** filter makes extensive use of this protocol, and provides support for status and error reporting. In cooperation with the **LPRng** print spooler, it will provide a detailed description of the actual print job progress and any error conditions that arise.

To use a Socket connection with **LPRng**, you use the `:lp=host%port` printcap entry shown below. The **lpd** print spooler will open a connection to the TCP/IP `port` on *host* and passes the (bidirectional) connection to the **ifhp** filter on file descriptor 1 (STDOUT) and the file to be printed on file descriptor 0

(STDIN). Errors and status information are reported by the **ifhp** filter on file descriptor 2 (STDOUT) and placed in the error status log by the **lpd** print spooler.

The connection made by the **lpd** server to the printer is *persistent* over the entire job; all file transfers for the same job are made over the same connection. This is important as it prevents other printer users from *hijacking* the printer in the middle of print a job and getting your job outputs mixed together.

The following is a typical printcap entry using the socket protocol.

```
raw:
  :lp=host%9100
  :sh:sf:mx=0
  :sd=/var/spool/lp
  :filter=/usr/local/libexec/filters/ifhp
```

4.5. Appsocket Protocol (TCP/IP) Operation

The Tektronics Phaser Series printers and QMS printers use the *Appsocket* protocol when sending a job to the printer. This protocol uses two ports: a TCP/IP *listening* port which accepts TCP/IP connections and a UDP *query* port that is used to obtain status information. Unfortunately, the UDP port is almost totally useless for job monitoring and status purposes and is not used except in an advisory role.

The Appsocket protocol is (briefly):

1. When a UDP packet is received on the UDP port a reply packet containing the status is returned to the originator's address. This packet contains an status indication in an *un-standardized* format but usually is readable or has a clearly defined format.
2. To send a job to the printer, a TCP/IP connection is opened to the TCP/IP port and job data is sent. Only a single job can be sent at a time - a EOJ in the job, i.e.- CTRL-D for PostScript or ESC E for PCL will cause the printer to terminate reading from the TCP/IP port, and after job processing has finished, to close the TCP/IP connection. All input after the EOJ may be ignored by the printer and not processed.
3. While processing the job, if *bidirectional* support is available and has been enabled the printer will return job status or information until all of the print job which is has received has been processed. This support is usually not enabled by default and must be enabled by using a specialized administration interface or configuration tool.
4. Unfortunately, some printers will also close the connection when the EOJ has been received. These printers are virtually useless when trying to get error or status information about a job.

5. Even more annoying is the behavior of some printers that insist on the network connection remaining open until the job has been processed. The device sending the job to the printer must do a *shutdown* on the sending direction of the network connection, and then read status information from the receiving direction until the printer terminates the connection. Unfortunately, some printers *do not* terminate the connection, and the the user must close the connection after a suitable timeout.
6. While processing the job, the printer will ignore any connection requests, and only until the job has been processed will the printer accept connections.
7. During job processing, status and error indications can be obtained by sending a query to the UDP port. However, the error conditions and other information are not very precise as the status may change dramatically during job processing.

The Appsocket protocol does not use a *persistent* connection. If two people are sending jobs to the printer simultaneously it is very likely that the jobs will get intermixed.

The `appsocket` option causes the **ifhp** filter to open and close a TCP/IP connection to the printer. In order to reopen the device, **ifhp** needs the device name. It gets this from the `PRINTCAP_ENTRY` environment variable which has the device information, or by using the TCP/IP address of the initial end of the connection. The following is a sample printcap entry for this printer:

```
# Phaser Setup
#  Appsocket
lp:server
  :lp=10.0.0.1%9100
  :sd=spooldir
  :...
  :ifhp=model=ps,appsocket
  #path to ifhp filter
  :filter=/.../ifhp
```

For your convenience, the `model=phaser` entry is suitable for use with the appsocket protocol.

4.6. Common Print Server Boxes Configuration Information

The following is a list of print server manufacturers, models, and with hints on how to access these boxes with various protocols.

Table 4-1. Network Print Servers

Manufacturer	Model	RFC1179 Port Name (rp=XXX)	Send to TCP port
Digital Products Inc. (http://www.digprod.com/)	NETPrint Print Server	`PORT`*n*, where *n* is port on server	- Unknown if supported -
Electronics For Imaging Inc. (http://www.efi.com/)	Fiery RIP i series	`normalq` or *urgentq*	- Unknown if supported -
	Fiery RIP XJ series	`xjprint`	- Unknown if supported -
	Fiery RIP XJ+ and SI series	print_*Model*, e.g. print_DocuColor	- Unknown if supported -
	Fiery models ZX2100, ZX3300, X2, X2e	`print`	- Unknown if supported -
Emulex Corp. (http://www.emulex.com/)	NETJet/NETQue print server	PASSTHRU	- Unknown if supported -
Extended Systems Inc. (http://www.extendsys.com/)	ExtendNet Print Server	*Printer*n, where *n* is port on server	- Unknown if supported -
Hewlett-Packard (http://www.hp.com/)	JetDirect interface card	`raw`	9100
	JetDirect Print Server	Port1=`raw1`, Port2=`raw2`, ...	Port1=9100, Port2=9101, ...
I-Data (http://www.i-data.com/)	Easycom 10 Printserver	`par1` (parallel port 1)	- Unknown if supported -
	Easycom 100 Printserver	`LPDPRT1`	- Unknown if supported -
IBM (http://www.printers.ibm.com/)	Network Printer 12, 17, 24, and 24PS	PASS	- Unknown if supported -
Lantronix (http://www.lantronix.com/)	EPS1, EPS2	`EPS_XXXX_S1 (serial) port 1, EPS_XXXX_P1 (parallel) port 2`, etc.	3001 (port 1), 3002 (port 2), etc.
QMS (http://www.qms.com/)	Various Models	RAW	35 (Appsocket)

Manufacturer	Model	RFC1179 Port Name (rp=XXX)	Send to TCP port
Tektronix (http://www.tek.com/color_prnters/) (Now Xerox (http://www.tektronix.xerox.com))	Tektronix printer network cards	PS (PostScript), PCL (PCL), or AUTO(Auto-selection between PS, PCL, or HPGL). Not reliable.	9100 (Appsocket on some models)
Rose Electronics (http://www.rosel.com)	Microserve Print Servers	lp	9100
Xerox (http://www.xerox.com/)	Models 4505, 4510, 4517, 4520	PASSTHRU	2501 (Appsocket on some models)
Model 4512	`PORT1`	10001 (programmable)	
Model N17	RAW	9100	
Models N24 and N32	RAW	2000	
Models 4900, 4915, 4925, C55	PS	2000	
Document Centre DC220/230	`lp`	- Unknown if supported -	

All company, brand, and product names are properties of their respective owners.

4.7. Timeout Problems Sending A Job

The **ifhp** filter may need to run a program such as `ghostscript` to do format conversion. For large files this can take quite a bit of time and most network printers have a *connection timeout*. If no data is received for this time the printer will close the connection. By default this timeout is fairly short: 30 or 90 seconds on most printers.

If you are sending large jobs to the printer using the `socket` protocol and are getting timeout problems due to conversion timeouts, then there are two solutions: a) use the Appsocket protocol, which will open and close the connection for each file, and only send data when the converted file is available, or b) do your conversions first and then spool the converted job to be sent directly to the printer. The second method requires an **LPRng** bounce queue.

```
# Method a) Appsocket
lp:server
  :lp=10.0.0.1%9100
  :sd=spooldir
```

```
  :...
  :ifhp=model=printer,appsocket
  #path to ifhp filter
  :filter=/.../ifhp

# Method b) Bounce Queue
#  this queue does the conversion if required
lp:server
  :lpd_bounce
  :lp=real@localhost
  :sd=spooldir
  :...
  :ifhp=model=printer
  #path to ifhp filter
  :filter=/.../ifhp
# this queue does transmission
raw:server
  :lp=10.0.0.1%9100
  :sd=spooldir
  :ifhp=model=printer
  #path to ifhp filter
  :filter=/.../ifhp
```

For method a), the Appsocket protocol is used and the **ifhp** filter will be invoked before sending a job. For method b), you use two queues: a *bounce* queue that does the format conversion and then sends the job to the real queue, and the real queue that actually talks to the printer.

4.8. PS, PCL, PJL Printer with TPC/IP Network Interface

The most common TCP/IP protocols used for transferring jobs to network printers are RFC 1179, a direct TCP/IP socket, connection to the print engine, and the very odd Appsocket protocol described in previous sections. Here is a reprise of the various printcaps and methods to use them.

```
# printer setup
#  force clients (lpr, lpq, to use server)
lp:lp=lp@serverhost
# server information
lp:server
  :sd=spooldir
  :...
```

```
# No filtering, transfer using RFC1179, use:
:lp=queue@10.1.1.1
#    or
:rp=queue:rm=10.1.1.1

# Filtering and then transfer using RFC1179, use:
:lpd_bounce:lp=queue@10.1.1.1
#    or
:lpd_bounce:rp=queue:rm=10.1.1.1
:ifhp=model=name
:filter=/.../ifhp

# Filter, transfer using socket, use:
:lp=10.1.1.1%9100
:ifhp=model=name
:filter=/.../ifhp

# Filter, transfer using Appsocket, use:
:lp=10.1.1.1%9100
:ifhp=model=name,appsocket
:filter=/.../ifhp
```

If your printer is a parallel port printer connected to an *external* Network Print Spooler such as an HP JetDirect box, then while the network connection to the Network Print Spooler is bidirectional the connection from the Network Print Spooler to the printer may be unidirectional and no status information will be returned from the Network Print Spooler. In this case you *must* add the `status@` option to tell **ifhp** not to expect status:

```
# Filter, transfer using socket
  :lp=10.1.1.1%9100
  :ifhp=model=name,status@
  :filter=/.../ifhp
```

4.9. PS, PCL, PJL Printer with Parallel Port Connection

If your printer is connected to a *bidirectional* parallel port you may be able to read status from the printer. First, determine if your printer has bidirectional IO capability and if your operating system has

support for it. If it does not, then do not use the `:rw` (open connection read-write) option to open the printer device in read write mode.

```
# printer setup
#  force clients (lpr, lpq, to use server)
lp:lp=lp@serverhost
# server information
lp:server
  # do now open read write
  :rw@
  :sd=spooldir
  :...
  # parallel port
  :lp=/dev/lpt
  #path to ifhp filter
  :filter=/.../ifhp
```

If, on the other hand, your operating system reports that the parallel port is bidirectional and is able to read the printer model information, then you can try opening the parallel port read-write and seeing if the **ifhp** filter can read status information:

```
# printer setup
#  force clients (lpr, lpq, to use server)
lp:lp=lp@serverhost
# server information
lp:server
  # open read write
  :rw
  :sd=spooldir
  :...
  # parallel port
  :lp=/dev/lpt
  #path to ifhp filter
  :filter=/.../ifhp
```

4.10. PS, PCL, PJL Printer with Serial Port

It is strongly advised that serial ports not be used for high speed data transfers. The main problem is trying to configure them in such as way that they do not lose characters due to data overruns or parity errors. LPRng is strongly deprecating support for serial port printers.

The **LPRng** print spooler will open and set the serial line characteristics, and pass the open connection to the **ifhp** filter. The `tty` connection must pass all 8 bits with no parity, and should use hardware flow control if at all possible. Unfortunately, the various `stty` options needed to do this vary from system to system. Also, you may discover that your serial connection does not support hardware flow control. If this is the case, then you will have to use software flow control which is rather unreliable for high speed (over 9600) serial lines due to the timing latencies involved.

```
# printer setup
#  force clients (lpr, lpq, to use server)
lp:lp=lp@serverhost
# server information
lp:server
  :sd=spooldir
  :...
  # serial port
  :lp=/dev/ttyxxx
  :stty=38400 -echo -crmod -raw -oddp -evenp \
     ixon pass8 -ixany cbreak crtscts
  #path to ifhp filter
  :filter=/.../ifhp
```

4.11. PostScript Only Printer

The `model=ps` entry supports PostScript only printers.

```
# printer setup
#  force clients (lpr, lpq, to use server)
lp:lp=lp@serverhost
# server information
lp:server
  :sd=spooldir
  :...
  :ifhp=model=ps
```

```
#path to ifhp filter
:filter=/.../ifhp
```

If you have a *unidirectional* or *write only* (no status information) connection such as a parallel port you should use:

```
:ifhp=model=ps,status@
```

Many PostScript printers recognize the PostScript EOJ marker (Control-D or `\004`) as an end of PostScript job indication and will perform page eject and other suitable actions. Unfortunately, strictly according to PostScript documentation this character is only allowed in the Serial Port Data Stream, and there are some printers that treat it as an error. In addition, the Control-T character is recognized as a printer status solicitation, and some printers do not return status or recognize it as an error.

If your printer does not handle PostScript EOJ (Control-D) at all, set `ps_eoj@` to suppress generation of extra Control-D characters by **ifhp**. If your printer requires a Control-D at the end of the job but fails when they occur at the start of the job, set `ps_eoj_at_start@`. If the printer requires a Control-D at the start but not at the end, set `ps_eoj_at_start@`.

PCL based printers are not nearly as fussy. However, you may discover that some of them do not correctly handle a PCL EOJ at the start of a job in spite of all examples and documention. Use the `pcl_eoj_at_start@` to suppress adding a PCL EOJ (Esc E) command string to the start of a PCL job file.

```
:ifhp=model=ps,ps_eoj@
```

See the section on File Conversion Support for ways to print text and other files on a PostScript printer.

4.12. GhostScript

Generating a raster image from a PostScript or PCL file in a timely manner requires a high speed processor and substantial amounts of memory. Many of the low cost printers require the user’s system to do the raster conversion and the raster file is then transferred to the printer. The file format is usually a subset of PCL.

The GhostScript program can process PostScript files and produce raster output for a wide range of devices. The `ghostscript` `pcl_gs` printer configurations is used with these printers. See GhostScript Printer for details.

4.13. Tektronics Phaser, QMS, and Appsocket Protocol

The Tektronics Phaser, QMS Network Printers, and a few others use the Appsocket protocol described in a previous section. The Tektronics (`model=phaser`) configuration entry has the required options for these printers:

```
[ phaser qms ]
appsocket
ps
pjl@
pcl
```

The following shows a typical printcap entry:

```
#  force clients (lpr, lpq, to use server)
lp:lp=lp@serverhost
# server information
lp:server
  :sd=spooldir
  :lp=10.1.1.1%35
  :...
  :ifhp=model=phaser
  #path to ifhp filter
  :filter=/.../ifhp
```

Chapter 5. Options and Arguments

- `model=`*Model Information*
- `model_from_option=`*Option with model information*

The **ifhp** filter is designed to work with the **LPRng** print spooler and expects to be passed the standard set of filter options. These have the form:

```
/.../ifhp [-c] [-X option]* accountingfile
Example:
/.../ifhp -n root -H hostname -P printer -s statusfile acct
# - X is any letter except T
```

All of the option letters except `T` are reserved by the **LPRng** program to pass information to the filter. For details about the options, please consult the **LPRng** documentation.

5.1. Command Line Options

The most important options that **LPRng** passes and that **ifhp** uses are:

-s statusfile

The file where **ifhp** status information is placed.

-Z useroptions

The `lpr -Z` options passed by the user, and are discussed in the options section.

-T options

These are usually options specified in the printcap entry and are discussed in the options section.

accountingfile

The file where accounting information is written.

`Examples:`

```
ifhp "-Tmodel=ps,status@" "-Za4,landscape"
```

Since commas are used to separate options, whitespace is used to separate multiple values for a particular option. You will need to quote this on a command line. For example:

```
ifhp "-Tfont=elite greek1 dingbat"
```

The **ifhp** program first checks to see if the PRINTCAP environment variable is defined. By convention, **LPRng** will place the printer printcap entry in this variable when it starts the **ifhp** filter. The printcap `:ifhp=options` value is extracted and used as the default `-T` options. After getting the options from the printcap, the `-Toptions` command line options are appended to the list of `-T` options. The single letter command line options are also made available to the **ifhp** programs as shown below:

```
PRINTCAP=lp:ifhp=model=this,status@:...

ifhp -n root -h localhost -Tmodel=that,debug=1

Concatenated -T options:  model=this,status@,n=root,h=localhost,model=that,debug=1
Resulting    -T options:  status@,n=root,h=localhost,model=that,debug=1
```

The `-T` option list is scanned from left to right, and later option values override earlier ones. The `-T` option values have priority over values that are obtained from the configuration file and cannot be overridden. There are several options that have important effects on the operation of the **ifhp** filter.

5.2. General Configuration Options - config, trace, debug

- `config=`*Configuration file location*
- `debug=`*Debug options*
- `trace` FLAG *trace on STDERR*

These options are used to control the global operation of the **ifhp** filter, and are only available from the `-T` command line options.

config=pathname

The `config` option specifies the location of the `ifhp.conf` file. This overrides the default location. The pathname can be a file name, list of filenames separated by spaces, or a filter. For example:

```
ifhp '-Tconfig=/usr/local/etc/ifhp.conf'
ifhp '-Tconfig=|/usr/local/bin/getconfig'
```

The second example uses the `getconfig` program to obtain configuration information. The configuration information is read from the program's STDOUT. There program is invoked with no command line options and is passed the environment variables that were provided to **ifhp**.

model=MODEL

The `model` option selects the portion of the **ifhp** configuration that will set values of configuration parameters. This is discussed in detail in the next section.

trace

As **ifhp** processes the print job, it produces tracing and error message information. By default this is written to the status file specified by the `-s` command line option. The `trace` option will cause this information to be written to STDERR (file descriptor 2). This is usually used in debugging.

debug=n

This option sets the debugging level to `n`, where *n* is an integer number. Level 0 turns debugging off, level 1 produces a small amount of verbosity and increasing levels produce more verbose information.

5.3. Status Messages

- `statusfile=`*statusfile*
- `statusfile_max=`*maximum status file size*
- `statusfile_min=`*minimum status file size*
- `summaryfile=`*one line summary file*

statusfile=pathname or -s pathname

The status file pathname is set by the command line `-s pathname`, or if it is not present then the `statusfile=pathname` configuration option. The file must exist and will not be created.

statusfile_max=n

If the status file is larger than statusfile_max K bytes (default 8K), then it is truncated to `statusfile_min=min` K bytes.

statusfile_min=n

The minimum size in Kbytes of the status file after truncation (default 1K).

5.4. Printer Status Available - status

- `status` FLAG *status available from device*

The `status` option indicates that there is a bidirectional connection to the printer, and that status can be obtained from the connection. During initialization the **ifhp** filter will test the printer connection and determine if it supports reading. If it does not then **ifhp** will set `status@`.

5.5. Monitoring Options - sync, waitend, pagecount

The `sync`, `waitend`, and `pagecount` options are ignored if no status is available from the printer. The `sync` option specifies the method to use to determine if the printer is ready and operational. The `waitend` option specifies the method used to determine when a print job is finished. The `pagecount` option specifies the method used to obtain pagecount or status information.

sync@, waitend@, pagecount@

This form of the tag indicates that the particular facility is disabled.

sync=pjl

PJL is used to determine if the printer is ready. This can be done by sending a PJL JOB or PJL ECHO command to the printer and waiting for return status.

sync=ps

A small PostScript job which causes a status report to be returned is sent to the printer.

waitend=pjl, waitend=ps

This is similar to the `sync` operation, but is done at the end of a job in order to determine if the printer is busy.

pagecount=pjl

Many PJL capable printers support reporting total page usage by means of PJL. This option causes a PJL command to be sent requesting the total page usage by the printer.

pagecount=ps

A small PostScript job which causes a status report to be returned is sent to the printer.

5.6. User -Z Option Support

The **ifhp** filter provides a simple way for users to request a particular printer facility or option. The `lpr -Zkey=value` command causes the **lpd** print spooler to pass the `-Z` options on the **ifhp** command line.

The **ifhp** filter implements these options by first determining if they are allowed, and then using them to select a set of strings that are sent to the printer. Since some options are implement by sending PJL strings to the printer, some by PostScript, and some by PCL commands, the method of specifying and generating them is a bit involved.

The following facility is used to control the names and types of user options.

pjl_user_opts=[...]

This tag specifies the list of user options that are implemented by sending PJL strings to the printer. This is available only if the printer is PJL capable.

pcl_user_opts=[...]

This tag specifies the list of user options that are implemented by sending PCL strings to the printer. This is available only if the printer is PCL capable.

ps_user_opts=[...]

This tag specifies the list of user options that are implemented by sending PostScript strings to the printer. This is available only if the printer is PostScript capable.

For each option, the actual string or set of strings is specified as follows.

pjl_*key*= ...

The value of the PJL user option `key`. This value can be one or more lines; the lines are checked for correct PJL format and sent to the printer before any language specific information.

ps_*key*= ...

The value of the PostScript user option `key`. This value can be one or more lines; leading and trailing whitespace is removed and the lines are placed before the first lines of a PostScript job file.

pcl_*key*= ...

The value of the PCL user option `key`. This value can be one or more lines; whitespace and new lines are removed and the characters are placed before the first characters of a PCL job file.

The following user options are predefined in the default `ifhp.conf` file and are recommended for use.

a3, a4, a5

Use a3, a4, or a5 paper

copies=N

Print N copies of a page or job

duplex

Use duplex printing, tumble on. Pages will come out so that the margins are at opposite ends of a page.

duplexshort

Use duplex printing, tumble off. Pages will come out so that the margins are at the same ends of a page.

envelope

Select envelope media

inlower

Select media from lower input bin.

inupper

Select media from upper input bin.

landscape

Use Landscape orientation

lduplex

Alias for duplex

ledger

Select ledger size (11x15 inches) media

legal

Select legal size (8.5x15 inches) media

letter

Select letter size (8.5x11 inches) media

manual

Select media from manual feed

mediaselect=N

Select media number N

outlower

Put output in lower tray or bin

outupper

Put output in upper tray or bin

oversize

Select oversize media

portrait

Use Portrait orientation

sduplex

Alias for simplex. Print on the single side of the media.

simplex

Print on the single side of the media.

transparency

Select transparency media

5.7. Adding User Options

The following shows how to add a PJL option to an `ifhp.conf` file. By convention, the configuration is added to the end of the `ifhp.conf` file.

```
[ newprinter ]
pjl_user_opts += [ screen ]
pjl_screen = PJL SCREEN = ON

ps_user_opts += [ fuzzy ]
ps_fuzzy = <</Fuzzy (\%s{fuzzy})>> setpagedevice
```

In the first example we define the `screen` option. The `lpr -Zscreen` option will cause the PJL command `PJL SCREEN = ON` to be put into the output to the printer.

Similarly, the `lpr -Zfuzzy=5` option will cause the PostScript command `<</Fuzzy (\%s{fuzzy})>> setpagedevice` to be sent to the printer.

5.8. Initialization and Setup Control

Several options are used during the processing steps discussed in Filter Operation Details to control what setup is done for the printer.

pjl_init = [...]

If PJL is enabled on this printer, options in this list are expanded and the resulting values are sent to the printer. After this, the -Z options are expanded and any options which are listed in the pjl_user_opts are processed.

ps_init = [...]

If PostScript is enabled on this printer and a PostScript file is being processed, then the options in this list are expanded and the resulting values are sent to the printer. After this, the -Z options are expanded and any options which are listed in the ps_user_opts are processed.

pcl_init = [...]

If PCL is enabled on this printer and a PCL file is being processed, then the options in this list are expanded and the resulting values are sent to the printer. After this, the -Z options are expanded and any options which are listed in the pcl_user_opts are processed.

These initialization options are very useful in order to set up information controlling the default format or options for a print job. For example:

```
pcl_init = [ normalpage ]
pcl_normalpage=[ letter crlf linewrap
  portrait clearmargins fixed pitch=10 courier ]
```

When processing a PCL job, `normalpage` is expanded by searching first for `normalpage` and then for pcl_normalpage; this in turn results in the expansion of the list of values. For example, pcl_crlf is usually defined as `pcl_crlf=\033&k2G`, which is the PCL command to translate a New Line (`\015`) character as a Carriage Return/New Line. The other entries have similar definitions that produce the desired effects.

Chapter 6. Configuration File

This section will cover the `ifhp.conf` file and the various options and configuration methods used to control the operation of the **ifhp** filter.

6.1. Configuration File Entries

The **ifhp** filter uses a simple text based configuration file, usually `/usr/local/etc/ifhp.conf` or `/etc/ifhp.conf` to get a set of configuration values which control its operation. The following sample configuration file segment shows how information is specified.

```
# comment line - first non-blank character is a #
#---- DEFAULTS ----
# we first have the default section
#   - a flag option whose value is 1
on_flag
#   - a flag option whose value is 0
off_flag@
#   - a flag option whose value is a string (single line)
#     its value will be 'this is a string'
strval = this is a string
#   - a flag option whose value is multiple lines
#     each additional line starts with whitespace
#     value is 'this\nis1\na\nstring'
longstrval = this
 is\061
 a
 string
#   - and a list that gets expanded -
#     '[ this ] [ is a\nlist ]' -> [ this is a list ]
longlist = [ this ] [ is a
 list ]
#    we can extend a string.
#    strval will  now be 'this is a string added'
strval += added
#    and we can expand a list
#     '[ this ] [ is a\nlist ] [ more ]' -> [ this is a list more ]
longlist += [ more ]

# a printer specific section
# ---- PRINTER ----
```

```
[ hp hp4* ]
# this match model=hp, model=hp4, model=hp4x
# override the default
onflag@
include /usr/local/etc/ifhp.conf.local

[ entry1 ]
value
[ entry2 ]
tc=entry1
```

6.2. Comments

Comments are lines whose first non-whitespace character is #. Use \# if the first non-whitespace character must be #.

6.3. Option Setting

```
Syntax             Equivalent To
option                     option=1
option@                    option=0
option=val
option=[ v1 v2  ... ]      value contains all whitespace
option=[ v1                up to the next option entry
 v2                        blank lines and comments
 v3                        are not included
 ]
option=v1
 v2
 v3
```

If an option's default value is the empty string (″). The **ifhp** program uses the Perl language convention that this value is equivalent to 0 when used in a numerical context or the empty string when used in a string context.

In general when a string is used in an integer context it is converted to a the appropriate numerical type using the standard Perl/C numerical representation and conversion methods.

The `flag` syntax sets the value of *flag* to the string `'1'`, that is, the string with a 1 value, and `flag@` sets it to `'0'`.

The `option = value` syntax sets the option value to a string. The string can extend across multiple lines. A line starting with a space has its value appended to the previous option with a new line (`\n`) separator.

As shown in the example, the `+=` operator is used to append to a string value. The `[ option option ...]` syntax is used to specify that the value is a list. Lists are used to specify a list of options which can be flags or string values. Lists have the property of *recursive evaluation* which means that the individual list items will be further processed during printing. This is discussed later in detail.

The `include` facility is currently deprecated, and may not be implemented in future releases. It will cause the specified file to be read and processed at that point in the configuration file.

6.4. Option Use

Options and their values are used to control printer operation. There are two types of options: those with a predefined or *builtin* meaning to the **ifhp** filter and those which have their values sent to the printer when appropriate. The builtin options are listed and their use is explained in later sections.

6.5. List Expansion

The **ifhp** filter configures a printer by sending the values of options to the printer or performing built-in operations. An option can have a flag, string, or list value.

A LIST value has the form `[ v1 v2 ... ]`. When a list value is to be sent to the printer each of `v1`, *v2*, etc. is expanded in turn and the corresponding string value or builtin action is carried out. If the string value of a term is itself a list, the list will be expanded in turn. Recursive list evaluation will result in an error. The following is an example of list expansion:

```
t1=[ p1 p2 ]
p1=this is
p2=[ p3 p4 ]
p3=a
p4=test
```

The option `t1` is expanded by expanding *p1* and then *p2*; The expansion of `p1` produces `"this is"`, and `p2` produces `[p3 p4]`. This list is then expanded to produce `"a"` and `"test"`.

Some LIST options are used in printer language specific contexts and their values are processed appropriately. For example, pjl_init=[...] specifies a set of initialization operations for PJL printers, and pcl_init=[...] is used to specify the initialization needed for PCL printing. The expansion of the LIST entries is done in the language specific context. For PJL this requires that the output be well formed PJL commands, and for PCL that all whitespace be removed.

The context dependent expansion is required because sometimes it is necessary to do operations both using PJL and PCL or PJL and PS combinations to ensure correct printer operation. During expansion the language name and an underscore is prefixed to the list entry name and this is used as the option name during expansion. If the prefixed name is not found then the unprefixed name will be used. For example, suppose that we have:

```
pjl_init=[ initstr test ]
pcl_init=[ initstr ]
pjl_initstr=@PJL ECHO YES
pcl_initstr=\033(*0V
test=DONE
```

When PJL initialization is being done and we want string values for the pjl_init LIST, we expand `initstr` and `test` in the pjl_ context. First a defined pjl_initstr value will be looked for and then a defined `initstr` value. Since there is a value of pjl_initstr it will be used.

Similarly we will check for pjl_test and `test` values. Since pjl_test does not have a defined value the `test` value DONE will be used.

When PCL initialization is being done and we want string values for the pcl_init LIST, then we expand `initstr` in a similar way, resulting in `\033(*0V`.

We can use the list entry `[ option=value ]` to temporarily specify the value of a variable which is then used during language specific expansion. For example, suppose that we have the following set of definitions:

```
pjl_init=[ initstr=testing ]
pjl_initstr=@PJL INIT=\%s%lcub;initstr%rcub;XQ
```

As discussed in the next section, the `\%s%lcub;initstr%rcub;` will cause the value for the *initstr* value to be substituted into the pjl_initstr string. How this is done is discussed in the section on String Escape Sequences.

6.6. String Escape Sequences

String values have a syntax similar to PERL or C. The \ (escape) character indicates the start of an escape sequence string. This has the syntax:

Standard Character Replacement

`\f`, `\r`, `\n`, and `\t` are replaced in turn by the ASCII characters FF, CR, NL, and HT whose values are 014, 015, 012, and 011 respectively.

Octal Character Replacement

`\nnn`, where nnn is 3 octal digits, is replaced by the corresponding character with the specified value.

Option Value Replacement

`\%format{option} OR \%format[option]`

The value of the option will be determined and replaced by a formatted string. The option value is determined by the following algorithm.

1. When expanding a list value, the `option=word` will push the `option=word` combination onto an evaluation stack, and then the `option` value is expanded in the current language context.
2. When starting a search for `{option}` in a language context lang_, the stack of list values is searched in oldest to newest order for a match for lang_option and then for `option`. The first one found is used as the option value.
3. After searching the evaluation stack for `{option}` and finding no match, the `-Z` command line option values are searched for a matching entry.
4. If none is found, then the `-T` command line option values and next the printer configuration will then be searched for lang_option and then for `option`. If no match is found, then the empty string will be the result if a string is wanted, or the value 0 if a number is wanted.
5. If the result of this lookup is a list, then the list will be expanded in turn, and the concatenating values of the expansion will be used.
6. When starting a search for `[option]` the `-T` command line options will be searched first and next the printer configuration will be searched for lang_option and then for `option`. If no match is found, then the empty string will be the result if a string is wanted, or the value 0 if a number is wanted.
7. ***help*** If the result of this lookup is a list If no match was found, then the search rules for `{option}` will be used, and the list expansion will be done as described above. If no match was found a null (empty string) value will be used.

Option Value Format

```
  %[-][0][length[.precision]][format]
  %d{1}   =>  '1'       %s{1}   => '1'
  %3d{1}  =>  '  1'     %3s{1}  => '  1'
  %03d{1} =>  '0001'    %-3s{1} => '1  '
  %4.2f{1} => '1.00'
Special Case Conversion:
  %s{ThisWord} => 'ThisWord'
  %U{ThisWord} => 'THISWORD'
  %M{ThisWord} => 'Thisword'
  %L{ThisWord} => 'thisword'
```

The format specifies how the value is to appear, and is similar to the printf format usage.

Depending on the format type, a value will be converted and used appropriately. The empty string or null value (″) will be treated as a *'0'* value when used in an numeric context.

The default format is %d, ie, \%{val} would be \%d{val}. The numerical formats supported are: %d, %o, %x, %X, %e, %f, and %g; The %s format use the string value of the result.

The `U`, `M`, and `L` have the same behavior as the `s` format, but the string value is then uppercased, lowercased and the first letter uppercased, and lowercased respectively. This allows various programs that have fussy requirement about the case of their options to be handled correctly.

6.7. Language Context and Value Expansion

The **ifhp** filter sends initialization and configuration commands to the printer. Depending on the type of language of a print file (i.e. - PostScript or PCL), different command formats would need to be used to implement different options. For example, to implement a *landscape* option for a PJL aware printer you would need to send the PJL command `@PJL SET ORIENTATION=LANDSCAPE`. For a PostScript printer you would need to send a very strange string which would depend on the actual printer mode.

Our language context also includes various checks for language specific dependencies. This section refers to material that is discussed in depth in later sections of this document, and on first reading may be

a little confusing. However, if you are not aware of some of these restrictions then much of the information in the configuration files may be very confusing.

6.7.1. PJL Language

A PJL command has the form `@PJL OPCODE ...`, and PJL commands must be sent as a block before any other commands. In order to assist with this, the **ifhp** filter provides the following assistance. When expanding a list value, each list entry is expected to form a well formatted PJL command.

1. Before sending any PJL command to the printer, the PJL Universal Exit Command (`\033%-12345X`) string is sent to the printer. This is automatically done if `pjl` is enabled for the printer.
2. The list item is expanded, and all value substitutions are done. Leading and trailing whitespace is removed, all characters are converted to uppercase, and a new line (`\n`) value is appended to the command.
3. Because not all printers support all PJL commands, the **ifhp** filter performs uses the pjl_only and pjl_except configuration lists to ensure that the options are allowed by the printer. The OPCODE must appear in the pjl_only list and not in the pjl_except list. For example:

   ```
   pjl_only = [ JOB SET STATUS ]
   pjl_except = [ STATUS ]
   ```

   The pjl_only indicates that the printer supports the PJL JOB, SET, and STATUS commands, but the pjl_except list removes the STATUS from this list. This means that only the JOB and SET commands will be allowed.

4. If the command is a SET command, then the PJL variable must appear in the pjl_vars_set list and not in the pjl_vars_except list.

   ```
   pjl_vars_set = [ PAPER SIZE ORIENTATION ]
   pjl_vars_except = [ PAPER ]

   @PJL SET SIZE=A4
   @PJL SET PAPER=LETTER
   ```

   In the above example, the `SIZE=A4` command would be allowed and sent while the `PAPER=LETTER` command would be rejected and not sent.

6.7.2. PCL Language

When sending PCL initialization strings to a printer, it is essential to send nothing that could cause a printable character to be sent before the actual file contents. Such output could cause the location and positioning of text to be altered in unexpected ways. To avoid this, the following steps are taken when expanding a list in a PCL language context.

1. Before any PCL string is sent to the printer, the PCL End of Job (`\033E`) string is sent to the printer.
2. All whitespace (blanks, tabs, etc) are removed from the string value.
3. Next, all escaped values are substituted. At this point you can *force* printable strings containing whitespace into the output by using the `\nnn` escape mechanism.
4. All list values are concatenated and then sent to the printer.

6.7.3. PostScript Language

The PostScript language processing is very minimal, as there are few problems sending PostScript to a printer.

1. Before sending any PostScript initialization strings, the PostScript End of Job indicator (`\004` or Control-D) is sent.
2. Strings are then expanded and the escape sequences are substituted.
3. Individual strings have a newline (`\n`) appended to them before being sent to the printer.

6.8. Printer Entries

The `ifhp.conf` file is divided into printer entries by `[ pattern pattern ...]` lines. Each pattern is glob matched against the `model` option value, and if the match is successful then the options on the following lines until the next printer entry header are appended to the specific printer configuration entry.

By convention, each configuration file is assumed to start with the header `[ default ]`, and the initial set of lines are used to set default values for the various **ifhp** options.

The algorithm for scanning the configuration files first sets the `model` value to *default*, and extracts the default information. It then sets the `model` value to the user specified value, and rescans the configuration file information.

If users need to add or modify the `ifhp.conf` file, then they should add their entries to the end of the file, and override any default options by specific values in their new entry. To aid with system configuration and maintenance, the distributed `ifhp.conf` file has the following text at the end of the file:

```
##### This is the end of the standard ifhp.conf file.
##### Add your local files after this
##### If you want to override some entries, simply change the names to
##### something different, i.e. hp4 hp4.old
##### Here is a script to do this and then append your local file to the
##### end of the ifhp.conf file:
#####
##### #!/bin/sh
##### for i in $* ; do
#####   perl -spi.bak -e 's/ $i / $i.orig /g' ifhp.conf
##### done
#####
##### sed -n -e '1,/XXX END XXX/p' ifhp.conf >ifhp.conf.new
##### sed '1,/XXX END XXX/d' ifhp.old >> ifhp.conf.new
#####
##### You can probably improve on this.
#####
#### XXX END XXX #####

# user adds new default values here for all printer entries
[ default ]
# set default value
pcl_option= \033test

[ mypcl_printer ]
# override default value
pcl_option=
```

6.9. Include Facility

The `include filename` facility is similar to the standard compiler file inclusion facility. The specified file or list of files separated by commas or whitespace will be substituted for the indicated line.

6.10. tc Entry Inclusion Facility

The `tc=entry` facilty is similar to the printcap `tc` facility used in the **LPRng** software other places. The specified entry or list of entries separated by commas or whitespace will be substituted for the indicated line.

Chapter 7. Filter Operation Details

The **ifhp** filter operates by first reading a configuration file to determine the type of printer it is working with, and then proceeds to carry out operations requested by the values of option variables passed on the command line or found in the configuration files. In normal operation, input is read from STDIN (file descriptor 0) and results written to STDOUT (file descriptor 1). Status reports are written to a status file or optionally to STDERR (file descriptor 2), together with any error messages or diagnostics.

In addition to normal operation the filter can run in the OF mode and act as a printer initializer and job terminator. This is discussed in detail in the **LPRng** documentation. When in the OF mode the two character sequence `"\031\001"` to the filter. will cause the filter to suspend itself by sending itself a SIGSUSP signal. The print spooler will detect this and then send job files to the same output device. After the files have been transferred the the filter will be restarted with a SIGCONT signal.

7.1. Filter Pseudo-Code

The details of the filter operations are described in the following *pseudo-code*. The sections marked with ### are discussed later in this document in detail.

`///` See: Options, Initialization and Setup

```
###+++ Initialization and Setup
// get ifhp information from PRINTCAP_ENTRY environment variable
if( PRINTCAP_ENTRY environment variable has a value ){
 split printcap information into printcap fields
 if( :ifhp=options,options is present in printcap ){
 split the options list and place in the Toptions list
 }
}
Add the -T command line options to the Toptions list
Add the -Z command line options to the Zoptions list
foreach option in -Toptions do
    if( option = "debug=level" ){
        set Debuglevel = level;
    }
    if( option = "trace" ){
        output error and trace on STDERR
    }
    if( option = "config=pathlist" ){
        set configuration pathlist = pathlist;
    }
    if( option = "model=name" and model not set ){
```

```
        set model = name;
    }
}
Read the configuration files from the config file list
Prepend each file with a [ default ] header

Scan the configuration files for [ default ] entries;
  later entry values will override earlier ones.

Repeat the scan, but this time search for [ model ] entries
  matching the specified model.

Put the command line options and -T options into configuration
  information, effectively overriding the information from the
  configuration files.

if( appsocket ) {
   Get the :lp=... entry from the PRINTCAP_ENTRY environment variable
   if( no information ) {
       use the getpeername() to get the TCP/IP address of the current
       connection.
   }
   if( no informtion AND no dev=... parameter ) {
       error!
   }
   close connection to printer and set -Tdev= device or IP Address
}

// open a connection to the printer if required
// usually only done when appsocket protocol is used
if( device specified using -Tdev=device ){
    close(1)
 // if device is host%port, we open TCP/IP connection
 fd = open(device);
 // Note - status  opens RW
 //        status@ opens WO
 dup fd to 1; close fd;
}

###---
```

/// See: Synchronization and Pagecount

```
######+++ Synchronization and Pagecount
```

```
if( status returned by printer and sync requested ){
 do{
 send command and wait for timeout;
    } while( no response );
 if( appsocket ){
 close and reopen TCP/IP connection;
 }
}


if( status and pagecount requested ){
    // pagecount has the form pagecount@ (none),
    //   pagecount=ps, pagecount=pjl, ...
    if( pagecount=language has value ) do {
 if( pagecount TRUE ){
 set pagecount= pjl or ps depending on availability
 }
        if( pagecount = pjl and PJL INFO available ){
           send PJL INFO PAGECOUNT command to printer
 } else if( pagecount = ps ){
           send PS program to printer
 } else {
 terminate with error;
 }
    } while( no pagecount response );
 if( appsocket ){
 close and reopen TCP/IP connection;
 }
}
###---
```

/// See: PJL Initialization

```
### PJL INITIALIZATION
if( PJL enabled ){
    language = "pjl_"
    foreach option in pjl_init=[...] {
       expand the option using the language value
       #+++ PJL OPTION ACTIONS +++
       if( option in pjl_vars_set=[ ... ]
         and option not in pjl_vars_except
         expand "@PJL SET OPTION=\%{option}"
         output = expanded string value
       } else {
```

```
            if( option value is a string ){
              output = expanded string value;
            }
          }
          // output has the form @PJL COMMAND ....
          if( COMMAND is in pjl_only=[ ... ]
              and not in pjl_except=[ ... ] ){
              send output to printer
          }
          #--- end PJL OPTION ACTIONS
       }
       if( !OF_mode ){
            foreach option in -Toption=value {
               if( option in pjl_user_opts ){
                   #+++ USER PJL OPTIONS
                   // join 'pjl_' and the option name
                   expand 'pjl_' . option
                   // perform PJL actions as above
                       #+++ PJL OPTION ACTIONS +++
                       ....
                       #-- PJL OPTION ACTIONS +++
                   #--- USER PJL OPTIONS
               }
            }
            foreach option in -Zoption=value {
               if( option in pjl_user_opts ){
                   // perform USER PJL actions as above
                   #+++ USER PJL OPTIONS
                   #--- USER PJL OPTIONS
               }
            }
       }
    }

    ###--- PJL INITIALIZATION
```

/// See: File Conversion Support

```
    // language is set to the type of job language
    // - PS, PCL, TEXT, RAW, UNKNOWN
    //  the first part of the job file is read and the filter takes
    //  a (wimpy) guess at the job file based only on the first couple
    //  of characters;  language is  be PJL, PS, or TEXT, or RAW
    //  This is the same algorithm as the UNIX FILE utility
```

```
language = default_language (from configuration);
if( command line -c (binary) option present ){
    language = RAW;
} else if( -Zlanguage=xxx option present ){
    language=xxx
} else if( forceconversion set ){
    use UNIX file utility to get file type
} else if( file is PS file ){
    language=PS
 if( file starts with PS EOJ (CTRL-D)
 and ps_eoj_at_start is clear ){
 remove the PS EOJ
 } else {
 send a PS EOJ first
 }
} else if( file is PCL file ){
    language=PCL
 if( file starts with PCL EOJ (ESC E)
 and pcl_eoj_at_start is clear ){
 remove the PCL EOJ
 }
}
if( file conversion table specified then ){
 look up file type in conversion table;
 if( conversion program specified ){
     run input through conversion program
 }
 set file type to output type
}


if( language = TEXT and PCL allowed ){
    language = PCL;
}

if( language not recognized by printer ){
    exit with error;
}

if( PJL ENTER supported ){
 use PJL ENTER command to select language;
 send nullpad NULLS to force full buffer condition
```

```
    }
```

/// See: Language Specific Initialization

```
    // LANGUAGE SPECIFIC INITIALIZATIONS
    if( language = PCL ){
        foreach option in pcl_init {
            ###+++ expansion
            do expansion similar to PJL OPTION actions
                using "pcl_" prefix for option lookup;
            ###---
        }
        if( not in OF_MODE ){
            foreach option in -Toption do {
                if( option in pcl_user_vars=[ ... ] ){
                ###+++ expansion as above
                ###---
            }
            foreach option in -Zoption do {
                if( option in pcl_user_opts=[ ... ] ){
                ###+++ expansion as above
                ###---
            }
        }
        remove whitespace and expand string results;
    } else if( language = PS ){
        ###+++ language specific actions as above,
          using the ps_ prefix for lookup
          allow only user option in the ps_user_opts list
        expand string results but do not remove whitespace
    }
```

/// See: File Transfer and Error Status Monitoring

```
    Transfer job to printer, reading error and other information
      back from the printer if enabled

    if( language = PCL ){
        send PCL End of Job
    } else if( language = PS ){
        send PS End of Job
    }
```

```
// job termination

######+++ Synchronization and Pagecount as above
finished = 0
while( waitend and not finished ){
 // timeouts and retries are done here
 if( time taken is too long ){
 give up and report an error
 }
 if( waitend with PJL ){
 wait for end of job using UINFO;
 } else if( waitend with PS ){
 send PostScript echo program to printer
 if end_ctrl_t then add ^T
 }
 wait for response
 if( response has end of job indication ) {
 finished = 1;
 }
}
if( pagecount ){
 if( appsocket ){
 close and reopen connection;
 }
 get pagecount using previously described algorithm
}

###---

exit
```

7.2. Options, Initialization and Setup

During the setup step, the **ifhp** system will extract command line options and scan configuration files for printer entries. These operations are covered in detail in other sections.

7.3. Languages Supported- pjl, pcl, ps, and text

- `pjl` FLAG *PJL Supported*
- `pcl` FLAG *PCL Supported*
- `ps` FLAG *PostScript Supported*
- `text` FLAG *Text Supported*

These flags set the languages that are recognized or processed by the filter.

7.3.1. pjl_job FLAG

- `pjl_job` FLAG *Send PJL Job and EOJ*

If PJL is enabled and the `pjl_job` flag is SET a PJL JOB and PJL EOJ command will be generated and sent to the printer at the job start and end respectively. The `JOB` command has the form:

```
@PJL JOB NAME = "..." [ START = nnn ] [ END = mmm ]
```

The START and END values can be specified by `-Zstartpage=nnn` and `-Zendpage=mmm` command line options. The EOJ command has must match the JOB command.

```
@PJL EOJ NAME = "..." [ START = nnn ] [ END = nnn ]
```

7.3.2. pjl_enter FLAG

- `pjl_enter` FLAG *Send PJL ENTER*

If PJL is enabled and the `pjl_enter` flag is SET, a PJL ENTER LANGUAGE = xx command will be generated when PCL or PS files are sent to the printer.

```
@PJL ENTER LANGUAGE = PCL
@PJL ENTER LANGUAGE = POSTSCRIPT
```

7.3.3. remove_pjl_at_start FLAG

- `remove_pjl_at_start` FLAG *Remove PJL code from begining of job*

Some printer drivers will prepend PJL code to the begining of print jobs. If the `remove_pjl_at_start` flag is SET, **ifhp** will remove this PJL code.

7.3.4. nullpad STRING

- `nullpad=` ephasis/null character count/

Some older model HP printers require receiving a large number of NULL (0) characters to force commands in the input buffer to be read. This can be done using the `nullpad` option. In practice, this has turned out to be largely irrelevant as most newer printers and network interface printers do not have this problem.

7.3.5. pjl_console FLAG

- `pjl_console` FLAG *printer console messages*

When this flag is set and PJL is available and the PJL RDYMSG command is supported, then a short message will be put on the console.

7.3.6. remove_ctrl STRING

- `remove_ctrl=`*remove control characters*

The remove_ctrl string option species a list of (control) characters that will be removed from PostScript jobs. This solves the problem of jobs with embedded Control-T or Control-C characters causing abnormal printer operation. For example:

```
remove_ctrl=CT
```

would cause Control-C and Control-T characters to be removed.

7.3.7. tbcp FLAG

- `tbcp` FLAG *Use TBCP protocol*

The `tbcp` flag can be specified as a user option as well as a configuration file option. If the file type is PostScript and this flag is set then the file is transferred using the Transparent Binary Communication Protocol. (See the Adobe PostScript Language Reference Manual for details on the protocol.)

At the start of the PostScript job, the sequence `\001` *M* is sent. Afterwards, all control characters in the set `0x01, 0x03, 0x04, 0x05, 0x11, 0x13, 0x14, 0x1C,` are replaced by the two character sequence `\001` `X+'@'` or `X+'\100'` or is sent. For example:

```
C\001\003   ->  \001\115\103\001\101\001\103 or \001MC\001A\001C
```

7.4. Synchronization and Pagecounts

- `accounting=` *accounting program*
- `accounting_info=` *accounting information*
- `pagecount` FLAG or Option *pagecounter available*
- `pagecount_interval=` *get pagecounter interval*
- `pagecount_timeout=` *get pagecounter timeout*
- `pagecount_start=` *get pagecounter at job start*
- `pagecount_end=` *get pagecounter at job end*
- `pagecount_ps_code=` *PostScript to get pagecounter*
- `sync` FLAG or Option *sync required*
- `sync_interval=` *do sync at interval*
- `sync_timeout=` *sync timeout*
- `wait_for_banner` FLAG *wait for banner page*

Many printers are able to provide status information back to the filter. It is assumed that in these circumstances file descriptor 1 (FD1) is *bidirectional* and status information can be read from it. When

the `status` option is TRUE, FD1 is readable and is a device or communications socket, then the filter assumes that it can read FD1.

Synchronization is usually done in order to ensure that a previously spooled job or printer action has completed correctly, and the printer is ready to accept a new job. It is usually carried out by sending a request to the printer to echo a string back to the filter. Clearly, if the printer cannot provide status or echo values back, then synchronization is impossible.

The value of the `sync` option determines if a PJL ECHO command or simple PostScript program is used. The PostScript program has the form:

```
\004%!PS-Adobe-2.0
( %%[ echo: TODSTR ]%% ) print () = flush
\004
```

where TODSTR is replaced with the current Time of Day.

To control obtaining synchronization, the `sync_interval=nnn` and `sync_timeout=nnn` options are used. The PJL or PS command is repeated at `sync_interval=nnn` second intervals; if nnn is 0, then it is sent only once. If synchronization is not obtained within `sync_timeout=nnn` seconds, then the filter exits with an error status. A 0 value or `sync_timeout@` disables timeouts.

When the **ifhp** filter is operating in OF mode and the `wait_for_banner` option is true, the filter will wait until it determines that the banner page has been completely printed before carrying out other filter functions.

Pagecounts are used to do accounting and report the number of pages used for a job. Most printers have a hardware based pagecounter mechanism whose value can be read by the appropriate PJL command or PostScript program. For example, if the PJL INFO command

```
@PJL INFO PAGECOUNT
```

is supported by a printer, the printer will return a status message containing the current pagecounter value. Printers that support PostScript may also be able to access the pagecounter value using a PostScript program. The exact details of the PostScript program vary from vendor to vendor and the `pagecount_ps_code=...` option specifies the PostScript program to use. For example:

```
pagecount_ps_code=
  /p {print} def ( %%[ pagecount: ) p
  statusdict begin pagecount end 20 string cvs p
  ( ]%% ) p () = flush
```

The **lpd** print server and the **ifhp** filter must act in coordination to do reliable pagecounting. The following options are used by the **ifhp** filter to assist with this:

Option	Purpose
`accounting=.../`	accounting program
`accounting_info=AnPR`	accounting information
`pagecount@`	do not get pagecounter value
`pagecount`	get pagecounter using either PJL or PostScript if available on the printer (default)
`pagecount=pjl`	get pagecounter using PJL
`pagecount=ps`	get pagecounter using PostScript
`pagecount_start`	get pagecounter at job start (default)
`pagecount_end`	get pagecounter at job end (default)
`pagecount_poll=N`	if nonzero, poll printer and conclude pagecounter value is nonzero when identical N times (default 1)
`pagecount_interval`	if polling more than once, then leave this interval (in seconds) between polls.
`of_options=...`	Use these option values when running in OF Mode

The following options are used in the **LPRng** printcap entry to assist with getting the pagecounter values:

```
lp:
  # run at job start
  :as=/.../accounting_at_start
  # run at job end
  :ae=/.../accounting_at_end
  # -a filter option value or last command line argument
  :af=/.../acct
  # default filter
  :filter=/.../ifhp
  # of filter - run before and after job, can be suspended
  # desperation flag for desperate situations
  #:suspend_of_filter@
  :of=/.../ifhp
  #options
  :ifhp=...,of_options=pagecount waitend
```

The `:as` program is run at the start of a print job, and is used to determine if the user has sufficient resourses to print a job. The `:ae` program is run at the end of a print job and is used to collect the accounting statistics. The **ifhp** filter will write accounting information to the accounting file specified by the command line `-a` option, or the last command line argument. When both the `:of` filter and normal filters are used together, the accounting information will be nested as shown.

```
Normal Mode:
  start                     '-qProcessID' '-pPagecounter' \
     '-tStartTime' '-Pprinter' '-Hhost' '-nuser' '-Raccntinfo'
  end '-bPages' '-Telapsed' '-qProcessID' '-pPagecounter'
     '-tEndTime' '-Pprinter' '-Hhost' '-nuser' '-Raccntinfo' \
     '-TElapsedTime'

OF Mode:
  filestart                    '-qProcessID' '-pPagecounter' \
     '-tStartTime' '-Pprinter' '-Hhost' '-nuser' '-Raccntinfo'
  fileend '-bPages' '-Telapsed' '-qProcessID' '-pPagecounter' \
     '-tEndTime' '-Pprinter' '-Hhost' '-nuser' '-Raccntinfo'


Sample Accounting File Entry:

  start '-q10699' '-p234' '-t2000-05-24-09:27:47.784' \
      -Plp -Hh4.private -npapowell
  filestart '-q10700' '-p234' '-t2000-05-24-09:27:47.784' \
      -Plp -Hh4.private -npapowell
  fileend '-b0' '-T1' '-q10700' '-p235' '-t2000-05-24-09:27:47.863' \
      -Plp -Hh4.private -npapowell
  end '-b1' '-T1' '-q10699' '-p235' '-t2000-05-24-09:27:47.863'
      -Plp -Hh4.private -npapowell
```

The format of the information written to the accounting file is controlled by the `accounting_info=AHPn` **ifhp** configuration value. If they are present, the specified **ifhp** command line flags are appended to the end of the standard accounting information. The `accounting=...` option specfies a program to run at then end of the job. This program has all of the accounting information passed as command line options. The program should exit with a 0 exit code; otherwise the results are undefined.

The printcap :suspend_of_filter controls how the **lpd** spooler manages the `of` filter. When a file is to be printed normally, a special two character suspend message (`\031\001`) is written to the filter STDIN. When the **ifhp** filter detects this string in the input it is required to suspend itself by sending itself a

`SIGSUSP` signal. The `:suspend_of_filter@` flag causes the **lpd** process to close the `:of` filter rather than suspending it, and to start a new `:of` filter process when it needs one. This option is used when there can be at most one process communicating with the printer, or when the **ifhp** filter must totally reinitialize the printer at job end.

The `pagecount` option controls if and how the pagecounter value will be fetched. Currently `pagecount=ps` (PostScript) and `pagecount=pjl` (PJL) are supported. The `pagecount` form will use PJL if it is available; otherwise PostScript if it is available. The `pagecount@` suppresses pagecount operation. The pagecount_start and pagecount_end flags control if the pagecounter will be obtained at the start and end of the print job.

One of the major problems with getting printcounter values is that the print job must be totally finished or at least have all of its pages run through the paper feed stream when the pagecounter value is reported. Unfortunately, most manufacturers do not provide accurate ways to coordinate the two activities. The `waitend` option is used to enable the **ifhp** filter to send special command sequences to the printer which will detect the true end of job, but this may not be possible on many printers.

The `printcounter_poll=N` (default 1) option provides a method to deal with these types of printers. Commands to get the printcounter value are sent to the printer, and repeated at printcap_interval second intervals until the printcounter value has been stable for `N` readings.

The PJL TEOJ (True End Of Job) command has been used with only limited success to force End of Job reporting only when the job has finished. This can be sent to the printer during PJL initialization by specifying it as one of the PJL initialization strings:

```
pjl_init=[ ... teoj ... ]
pjl_teoj=@PJL TEOJ=ON
```

The of_options are used to modify the actions of the **ifhp** filter when it is running in OF Mode.

The pagecount_start and pagecount_end (both default to TRUE or ON) control if pagecounter values are obtained at the start or end respectively of the job.

The pagecount request is sent to the printer every `pagecount_interval=nnn` second intervals; if nnn is 0, then it is sent only once. If no pagecount value is obtained within `pagecount_timeout=nnn` seconds then the filter exits with an error.

7.5. PJL Initialization

- `pjl_init=[ ... ]` *pjl initialization*

- `startpage=` *start page to print*
- `endpage=` *end page to print*
- `pjl_only=[ ... ]` *pjl commands supported*
- `pjl_except=[ ... ]` *pjl commands not to be used*
- `pjl_vars_set=[ ... ]` *pjl variables supported*
- `pjl_vars_except=[ ... ]` *pjl variables not to be used*
- `pjl_user_opts=[ ... ]` *pjl options available to user*

If a printer supports PJL, the many printer operations can be initiated and controlled using PJL commands. Unfortunately, not all printers support the same set of commands. In addition, not all printers support the same set of operations or options. A PJL command has the form:

```
@PJL COMMAND OPTION OPTION ...
```

A PJL variable is set using:

```
@PJL SET var = value ...
```

The `pjl_only=[ ... ]`, `pjl_except=[ ... ]`, `pjl_vars_set=[ ... ]`, and `pjl_vars_except=[ ... ]` options are used to control which PJL commands and which PJL variables can be set. The pjl_only variable lists the commands supported by the printer, and the pjl_except lists commands *not* supported by the printer. Before sending a PJL command, the **ifhp** filter checks to make sure that the command name is in pjl_only and not in pjl_except. If the tests fail, then tne command is not sent.

Similarly, when sending a command to set a PJL variable, the pjl_vars_set and pjl_vars_except lists are checked to determine if the variable name is in pjl_vars_set and not in pjl_except list. If the tests fail, then tne command is not sent.

If PJL is enabled, then the following actions are taken.

1. PJL Universal Exit Language (UEL) `\033%-12345X` is sent to the printer.

   This is required to ensure that the following PJL commands are accepted.

2. PJL JOB command is sent at the start of job. The JOB command can be used to select pages or impressions to be printed. If the `-Zstartpage=nnn` or `-Zendpage=mmm` option is present, then the PJL JOB command has the form:

   ```
   @PJL JOB START=nnn END=mmm
   ```

3. The `pjl_init=[ ... ]` value option is expanded using the PJL (`"pjl_"`) language context as described above.
4. The `-Toption=values` and `-Zoption=values` are scanned for matching option names in the `pjl_user_opts=[ ... ]` list. If they are found, then the options are recursively evaluated in the PJL language context. The expansion algorithm will cause the option value to be used to set PJL variables. For example:

```
Configuration:
  pjl_vars_set=[ OUTBIN AUTOSELECT JAM=YES ]

Command
  ifhp -Zoutbin=upper,autoselect,jam

PJL command generated:
  @PJL SET OUTBIN=UPPER
  @PJL SET AUTOSELECT=ON
  @PJL SET JAM=YES
```

7.6. File Conversion Support

- `forceconversion` FLAG *force use of file utility*
- `default_language=` *default job language*

The `lpr -l` or `lp -b` flags indicate that the spooled files are not to be processed by an output filter. The **LPRng** spooler recognizes this option and passes the `-c` command line option to suppress any language specific processing for files.

However, many PostScript printers cannot handle text files, and produce many hundreds of pages of garbage output if they are sent to the printer without being translated into PostScript, and some printers require language specific setup in order to print PCL, PostScript or text files correctly.

The **ifhp** filter has builtin tests for PJL, PCL, and PostScript files. These tests are almost identical to those used by many printers which do *autodetection*. If you need to recognize a wider range of file types, you can configure **ifhp** to use the UNIX `file`(1) program.

Finally, some printers have a very specialized job format that requires conversion to by a `rasterizer` program. This is handled as detailed in the following sections.

7.6.1. File Type Detection

- `file_util_path=` *file utilty path*
- `forceprocessing` FLAG *force file processing*

The **ifhp** filter has a set of built-in tests to determine if the input job file is PJL, PostScript, and PCL, or (default) text, and flags the file with language types `pjl`, `ps`, `pcl`, and `text` respectively.

You can also use the UNIX file utility utility to determine type as well. The `file` utility is invoked with it STDIN attached to the file and **ifhp** uses the information it writes to STDOUT as the raw file type. The **ifhp** program will convert the output to lowercase, remove multiple whitespace characters, and replace the remaining whitespace characters with underscores _.

By default, **ifhp** will not try to detect *binary* files, i.e. - files printed with the binary or literal flag (`-c` command line flag). You can use the `forceprocessing` flag to cause all files of all types to be processed. **ifhp** will first try to use its builtin tests and then will use the `file` utility. You can set the forceconversion flag to force **ifhp** to only use the `file` utility. The following shows the information in the `ifhp.conf` file used to configure the file type detection.

```
##forceprocessing - check all files for type - default 'no'
forceprocessing@
## default
default_language=text
## force only use of file program
## default is to let ifhp try first, then try file
forceconversion@
##  file utility path
file_util_path=/usr/bin/file -
```

The output of the file utility is converted to lower case and used as the language type for further processing.

7.6.2. Conversion

- `file_output_match=` TABLE *file utility output match list*
- `language=` *language type override*

Once the language type has been determined, the **ifhp** filter then decides if a conversion program needs to be run and will convert the input file to a required file type. This activity is controlled by the file_output_match table.

```
file_output_match = [
  *postscript*  ps  \%s{ps_converter}
  *pcl*         pcl  \%s{pcl_converter}
  *pjl*         pjl  \%s{pjl_converter}
  *printer*job*language* pjl
  *text*  pcl  \%s{pcl_converter}
  *gzip_compressed*  filter  \%s{gzip_decompresser}
  ]
```

Each line of the file_output_match table contains a (URL encoded) *glob* pattern, the language type (`ps`, *pcl*, etc.) produced by the conversion program, and the (optional) conversion program.

The format:

```
file_output_match = </pathname
```

will cause **ifhp** to open and read the specified file for the file_output_match table. The file's contents must have the same format as the file_output_match table but without the `[` or `]` delimiters.

The file_match_table is scanned from first to last entry for a `glob` pattern that matches the file type determined by the **ifhp** program or the output of the file_util_path program. If no match is found, then the language is set to the default_language value.

The output language of the conversion program is set to the second entry. The conversion program will be run with its STDIN set to the input file and its STDOUT used as the converted output. If there is no conversion program then the original file is used and only the language type is modified.

The `filter` language type causes the specified conversion program to be run and then the output of the program to be reprocessed. As shown above, this allows file decompression routines to be used to expand the files.

The following are some short samples of what can be done with the conversion facility.

```
# set up GhostScript
gs_device=epsonc
gs_options=-r1440
gs=/usr/bin/gs
gs_converter= [ \%s{gs} -dBATCH -q -sOutputFile=- \
  -sDEVICE=\%s{gs_device} \%s{gs_options} - ]
# use GhostScript for conversion
```

```
ps_converter = [ \%{gs_converter} ]
text_converter= [/usr/bin/a2ps -q -B -1 -M Letter \
  --borders=no -o- \%s{ps_converter} ]
gzip_decompresser = [ /usr/bin/gzip -c -d ]

[ ghostscript gs ]
file_output_match = [
  *postscript*  ps  \%s{ps_converter}
  *text*  pcl  \%s{pcl_converter}
  *gzip_compressed*  filter  \%s{gzip_decompresser}
  ]

Printcap entry:

pr:
  :ifhp=model=ghostscript,gs_device=laserjet,gs_options=-r300x300
```

In this example, we have shown a very interesting device - the GhostScript device. We use the `gs` (GhostScript) program to do the conversion, and specify the gs_device and gs_options values in the printcap entry. There are a few details that should be observed when using this facility.

1. The output from the `file` program has spaces converted to underscores. If you need to match spaces then use the underscore in the pattern. For example, `ascii text` would be matched by **ascii_text**.
2. If the conversion program contains a shell meta character such as `|`, `;`, `>`, `<`, etc, then it will be executed using `/bin/sh -c 'command'`. This allows a pipe of conversion commands to be constructed. This is discussed in detail below.
3. The most commonly used conversion programs are GhostScript, used to convert PostScript to a format compatible with a non-PostScript printer, and the a2ps, enscript, and textps Text to PostScript conversion programs which convert text into PostScript for a non-text supporting printer. The use of a `wrapper` program with these utilities is discussed below.
4. All of the command line options can be substituted on the command line using `\%{X}`, where *X* is the single letter command line option flag.
5. The `\%s{ARGV}` value is replaced by the command line arguments.
6. The conversion program must exit with a 0 error code or an error is assumed to have occurred.

The `enscript` program will exit with a non-zero error codes even for successful conversions and we need to use a `wrapper` script that will run it and then return the correct error code as shown below.

```
#!/bin/sh
# /usr.../wrapper path [options]
# wrapper script for a2ps, enscript and others
#   path is the path to the program and options are the
#   options to pass.  The program is run and then the exit
#   code is corrected
"$@"
status=$?
case "$status" in
 1 ) exit $status ;;
esac
exit 0
```

7.6.3. LF to CR/LF Conversion

- `crlf` FLAG *LF to CR-LF conversion*

When processing `text` or *pcl* files, the `crlf` option will enable translation of LF (`\n`) to CR/LF (*\r\n*) sequences. If you are using **ifhp** to simply do LF to CR/LF translation, then you can use:

```
ifhp -Tcrlf
```

7.6.4. Text Treated Like PCL

Text is simply PCL with no special formatting codes. However, you will still need to send the PCL initialization strings to the printer. You can do this by using the following entry in the file_output_match table:

```
file_output_match = [
 *text*  pcl
 ]
```

7.6.5. Default to Passthrough

Your printer may be capable of handling a wide variety of job formats. If you want to simply pass through files of unknown type or language then use the following entry in the file_output_match table:

```
file_output_match = [
 *  raw
 ]
```

7.7. GhostScript Printer

Generating a raster image from a PostScript or PCL file in a timely manner requires a high speed processor and substantial amounts of memory. Many of the low cost printers require the user's system to do the raster conversion, and a raster file is then transferred to the printer. These files are usually in a proprietary format. The GhostScript program can process PostScript files and produce raster output for a wide range of devices. See the GhostScript documentation for details. Some printers have PCL support but do not support PostScript. The `gs` and `pcl_gs` printer configuration support these printers.

```
# PRINTER ghostscript - Printer with GhostScript conversion to raster files
gs_converter= [ /usr/bin/gs -dSAFER -dBATCH -q
   -sOutputFile=- -sDEVICE=\%s{gs_device} \%s{gs_options} -
 ]
text_converter= [ /usr/bin/a2ps -q -B -1 -M Letter --borders=no -o-
 ]

[ ghostscript gs ]
pcl@
pjl@
ps
text@
file_output_match = [
# PostScript to Raster
  *postscript*  raw  \%s{gs_converter}
# text to PostScript to Raster conversion
  *text*  filter  \%s{text_converter}
  ]

# PRINTER pcl_gs - PCL Printer with GhostScript conversion to raster files
```

```
[ pcl_gs ]
pcl
pjl@
ps
text@
file_output_match = [
# PostScript to Raster
  *postscript*  raw  \%s{gs_converter}
  *pcl*  pcl
# text to PostScript to Raster conversion
  *text*  filter  \%s{text_converter}
  ]
```

The `\%s{gs_device}` and `\%s{gs_options}` parameters can now be specified in the printcap. The following shows a typical printcap entry for use with this entry.

```
#  force clients (lpr, lpq, to use server)
lp:lp=lp@serverhost
# server information
lp:server
  :sd=spooldir
  :lp=/dev/lpt0
  :...
  :ifhp=model=gs,gs_device=epson,gs_options=-r240x72
  #path to ifhp filter
  :filter=/.../ifhp
```

The **ifhp** configuration entry uses GhostScript to do the rasterization of the PostScript file, and the `a2ps` program to do a text to PostScript conversion.

7.8. Language Specific Initialization

- `ps_init=` *PostScript initialization steps*
- `pcl_init=` *PCL initialization steps*

After determining the output file language type, language specific operations are then carried out by expanding the language`_init=[ ... ]` options in the language context, and then the options in the

`-Toption=value` and `-Zoption=value` command line options. The `-T` options are expanded before the `-Z`, allowing the `-Z` actions to override any set by the `-T` actions.

As mentioned elsewhere, the reason for the language specific processing is to allow different actions for the same command line option, depending on the file type that is being processed. For example, when processing a PCL file it might be necessary to send PCL command strings and when processing a PostScript file, you would need to send PostScript commands.

7.9. File Transfer and Error Status Monitoring

- `logall` FLAG *log all printer status*
- `pjl_error_codes` TABLE *PJL error code translations*
- `pjl_quiet_codes` TABLE *ignore these PJL error codes*
- `pjl_alert_codes` TABLE *alert operation on these PJL error codes*
- `pjl_alert_handler` TABLE *program to alert operator*

If the printer can return status, i.e., the `status` option is on, the filter will read status information from the printer.

If the `logall` flag is SET, then all error messages will be written to the status or log file.

If the printer is returning PJL status information, then this has a specific format:

```
@PJL UINFO DEVICE
CODE=nnnn
DISPLAY="value"
...

@PJL UINFO JOB
START
...

@PJL UINFO JOB
END
...
```

The **ifhp** program will extract the CODE and job start and end flags, and log these as appropriate.

Unfortunately, some PJL based printers are extremely verbose in their generation of status messages. In order to reduce the amount of logging of redundant information, **ifhp** will only record when a device status has *changed*, rather than when it has been reported.

The pjl_quiet_codes=[code code code] value is used to suppress reporting of selected error codes. If the error code is in the pjl_quiet_codes list, then the error status will not be reported to the user unless the `logall` option is set. The list of values is used as *glob* patterns. For example:

```
pjl_quiet_codes=[ 10000 10001 10003 10023 10024 35078 41* ]
```

Also, there may be error codes which do not have a builtin error message available. New messages can be added using the pjl_error_codes option. Its value is a list of lines, each line consisting of an error code followed by the corresponding error message:

```
pjl_error_codes=[
   code=msg
   code=msg
   ...
]

Example:
  pjl_error_codes=[
     10000=powersave mode
     10001=Ready Online
     10002=Ready Offline
     10003=Warming Up
     10004=Self Test
     10005=Reset
  ]
```

Finally, there are codes or classes of codes that require operator intervention. These are specified using the pjl_alert_codes, and the pjl_alert_handler program will be invoked to send them to the appropriate destination. The formated error message will be available on STDIN for the handler. For example:

```
pjl_alert_codes=[ 41* 42* 23* ]
pjl_alert_handler=/usr/local/libexec/filters/alert_handler
```

7.10. End of Job

- `waitend` OPTION *wait for job completion status*
- `waitend_interval=` *query for end at this interval*
- `waitend_ctrl_t_interval=` *send CTRL-T at this interval*

The `waitend` option controls the job termination sequence. By default, this will do the same work as the `sync` operation, and the option takes the same set of values.

If `waitend` is suppressed using `waitend@`, then as soon as a job has been transferred, the next step, `pagecount`, will be attempted. If the print job has not finished at this point, then erroneous page counts will be reported.

When the `appsocket` protocol is used, suppressing `waitend` will cause no error messages from the printer to be reported.

Some printers do not have a True End Of Job reporting capability using PJL. This means that the job will be reported as done, but paper is still moving through the print engine. If you try to get pagecounts at this point you will get the wrong value. An alternative method is to set `waitend=ps` and The `end_ctrl_t=word:word:...` This will cause a CONTROL-T to be sent to the printer, a PostScript convention that will cause the PostScript interpreter to return the actual printing status. In most printers this will be `printing` or something other than `idle` or *busy* as long as paper is moving in the print engine. When status is returned, the words in the `end_ctrl_t=word:word:...` list value are examined for a match. If the status word is present then the end of job condition is assumed.

The waitend_interval value controls how often the waitend operation is repeated. This is usually set to a fairly large value, as it is normally used only to recover from printer failures such as users turning the printer on and off.

The waitend_ctrl_t_interval controls how often the printer is queried for status using CTRL-T and is usually set to a short (2 or 3 second) value.

7.11. Tektronix Phaser, QMS and AppSocket Support

- `appsocket` FLAG *connect to printer and use AppSocket Protocol*

The `appsocket` flag is used to specify that data transfer will be done using the AppSocket protocol. The **ifhp** filter will open a connection to the ip address and port specified by the `dev=host%port` option and carry out the various operations that it needs to do.

Normally after sending information to the printer a `close()` operation is done on the network connection. This will usually be acceptible as the printer will print the job that was sent to it. However, some printers require that the network connection be left open to report status.

If page count information is needed, the **ifhp** filter will then reopen the connection and get the page count information.

Chapter 8. Banners and OF Mode Operation

One of the more difficult administrative issues is whether to print banners (job separators) or to save the large amount of wasted paper, time and effort. The **LPRng** and **ifhp** combination provide a rather esoteric set of methods to generate banners, at least one of which should be suitable for your application.

You should be aware that some printers have the obnoxious habit of generating their own banner pages when jobs are transferred via the RFC1179 protocol. You should consult the manufacturers documentation and take the necessary steps to turn printer banner page generation off. It may turn out that this is impossible to do, and in that case the only option is to use the Socket or Appsocket connection methods.

In the original BSD print spooler, the `:of` print filter was responsible for banner generation. The print server would send it a *magic string* that the filter would recognize as a *print a banner* request, and it would then generate a banner. After this, another special `magic cookie` string, the character sequence `0x19,0x01`, was sent to cause the `:of` filter to suspend itself. This allowed the print spooler to hold the connection to the printer open while it started another filter to transfer a file. Finally, after all the jobs were sent, the `:of` filter was sent a SIGCONT signal to wake up and perform whatever closing operations were necessary.

This mixing of functionality causes horrible problems when special setup strings must be sent to printers in order to configure them for various operation. In order to avoid these problems, the banner printing and filter functions have been separated in the **ifhp** filter. If a banner is needed, then **LPRng** will use a special banner generating program to generate the banner. The output of this program is then sent to the `:of` filter, or directly to the printer if there is no `:of` filter.

8.1. No Banner

Here is a typical printcap entry when banner printing is not wanted:

```
lp:
  :sh
  :filter=/.../ifhp
```

The `:sh` (suppress headers or banners) explicitly disables banner printing, and the **lpd** server will not even attempt to print a banner.

8.2. Banner Printing and No OF Filter

This printcap entry specifies a banner generator program and and banner generation. There is no `:of` filter specified, so the banner is sent directly to the printer. In such a case the banner printing program should make sure that it generates output with the appropriate set of initialization strings. The `pclbanner`, `psbanner`, and `lpbanner` programs produce PCL, PostScript, and text banners suitable for a wide range of printers. The code for these banner generators is part of the **LPRng** distribution.

```
lp:
  :bp=/.../pclbanner
  :of=/.../ifhp
  :filter=/.../ifhp
   # or
  :bp=/.../psbanner
  :of=/.../ifhp
  :filter=/.../ifhp
   # or
  :bp=/.../lpbanner
  :of=/.../ifhp
  :filter=/.../ifhp
```

8.3. Banner Printing With OF Filter

Finally, we may want banner printing together with the `:of` filter. This is usually the case when we need to perform special printer setups or require the **ifhp** filter to do accounting. In this case we need to make sure that the banner page is counted as part of the job:

```
lp:
  :bp=/.../pclbanner or :bp=/.../psbanner or :bp=/.../lpbanner
  :of=/.../ifhp
  :filter=/.../ifhp
```

When invoked as an `:of` filter, the **lpd** server passes a `-Fo` option on the command line, so that the **ifhp** filter knows what mode it is operating in.

8.4. LPRng Options Controlling Banner Printing

The `lpr -h` (no header or banner) option suppresses putting *banner name* (`L` line) into the control file. The **LPRng** :ab (always banner) overrides this, and will print a banner using the user name information. The `:sh` (suppress header) option will override everything and prevent banners from being generated.

Banners are generated by the `:bp=/...` (banner program) program; different banners at start and end can be generated by using the `:bs=/...` (banner start) and `:be=/...` (banner end) options, which override the `:bp=` option.

8.5. of_options option

This option, usually used on the command line or in the `printcap` file as part of the `:ifhp` information, specifies a set of options to use when the filter is running in OF mode. For example:

```
lp:
  :ifhp=sync@,pagecount@,waitend@,of_options=sync pagecount waitend
  :filter=/usr/local/libexec/filtes/ifhp
  :of=/usr/local/libexec/filtes/ifhp

same as:

lp:
  :filter=/usr/local/libexec/filtes/ifhp -Tsync@,pagecount@,waitend@
  :of=/usr/local/libexec/filtes/ifhp -Tsync,pagecount,waitend
```

In this printcap entry, the **ifhp** filter is invoked normally with the `sync@`, `pagecount@` and `waitend@` options. This causes it to send jobs as fast as possible to the output device and only monitor the device for error status. The `of=` entry causes the **lpd** print spooler to start **ifhp** in OF mode, and the of_options are used to override the other ones.

Chapter 9. Font Download Support

For historical reasons, there is support for downloading a font or other file to the printer. A large amount of the necessary operations are now in the `ifhp.conf` file.

The font_download built-in option supports downloading as described below.

9.1. PCL Font Downloading

The following shows the a typical `ifhp.conf` file which has PCL font downloading enabled.

```
#
# Fonts and Font Downloading
#  fontid is used to set the current font
pcl_init=[ ... font ... ]

# combination command
pcl_font=[ delete_fonts font_id font_download font_primary ]

# font control
#
font_op=0
pcl_font_op=\033*c\%{font_op}F
pcl_delete_fonts=\033*c0F

font_id=1
pcl_font_id=\033*c\%{font_id}D

# set primary font
font_primary=1
pcl_font_primary=\033(\%{font_primary}X

# font directory
pcl_fontdir=/usr/local/lib/fonts

#default font file
font=c1201b.10
```

To allow users to download a font and have it set up for PCL use, the pcl_init option should include the `font` option in an appropriate position in the initialization sequence. As shown above, this will get

expanded into the pcl_delete_fonts, pcl_font_id, pcl_font_download (which is has built-in support), and the pcl_font_primary options, which are expanded in order.

The pcl_font_download is supported by the builtin operation which will find the pcl_fontdir directory value and a value for the `font` variable, using values from the `-Z` and `-T` and configuration information in that order. If no `font` value is found, no font will be downloaded. For example:

```
lpr -Tfont=font1,font2
```

When the pcl_font_download option is expanded, it will generate the pathnames `/usr/local/lib/fonts/font1` and `/usr/local/lib/fonts/font2`, open these files, and send their contents directly to the printer.

9.2. PS Font Downloading

PostScript font downloading is supported in a similar manner to PCL font downloading.

```
#
# Fonts and Font Downloading
#
ps_init=[ ... font ... ]

# combination command
pcl_font=[ font_download ]

# font directory
ps_fontdir=/usr/local/lib/fonts

#default font file
font=font.ps.10
```

In a similar manner to the PCL font downloading, when the ps_init list is expanded, the ps_font entry will be expanded in turn. If the `-Zfont=ZapDingbat.ps` is specified, then the `/usr/local/lib/fonts/ZapDingbat.sp` file will be opened and downloaded to the printer.

9.3. PJL File Downloading

In a similar manner to the above font downloading, you can specify a configuration or other setup file that should be sent to the printer as part of the PJL setups. The following configuration shows how to set this up.

```
#
# PJL Initialization File Downloading
#  fontid is used to set the current font
pjl_init=[ ... setup  ... ]

setup=initval
font=\%s{setup}
# setup directory
pjl_fontdir=/usr/local/lib/fonts
pjl_setup=[ font_download ]
```

The above configuration will cause the value of the `setup -Z`, `-T` or configuration option to be used.

Chapter 10. Debugging and Problem Solving

If you are reading this section, then most likely you have had a problem using **ifhp** with **LPRng**. Before going any further, please read RFC1179 (BSD or TCP/IP) Job Transfer Printcap Entry, and make sure that your printcap entry has the lpd_bounce flag set. This is the *Most Frequently Asked Question* and the *Most Frequently Provided Answer* on the **LPRng** mailing list.

The following section outlines a method to debug problems with the **ifhp** filter. It will make use of some `diagnostic` options that are normally not used in a printcap entry. First, let us start with a typical printcap entry and a problem.

```
lw4:
  :lp=10.0.0.14%9100
  :sd=/var/spool/lpd/%P
  :ifhp=/usr/local/lib/filters/ifhp

Command:
  lpr -Plw4 -V /etc/motd

LPR output:

sending job 'papowell@h4+223' to lw4@localhost
connecting to 'localhost', attempt 1
connected to 'localhost'
requesting printer lw4@localhost
sending control file 'cfA223h4.private' to lw4@localhost
completed sending 'cfA223h4.private' to lw4@localhost
sending data file 'dfA223h4.private' to lw4@localhost
completed sending 'dfA223h4.private' to lw4@localhost
done job 'papowell@h4+223' transfer to lw4@localhost
Version LPRng-3.6.14
```

When trying to print to the print queue, the user discovers that no output comes out of the printer. The immediate suspicion is that the filter is not working. You should make sure that the `lpr` command is actually sending the job to the print queue. You can then use `lpq` to discover what happened:

```
# lpq -llll
Printer: lw4@h4  'Hp : Laserwriter'
 Queue: no printable jobs in queue
 Status: subserver pid 27251 starting at 15:34:09.350
 Status: accounting at start at 15:34:09.357
```

```
Status: opening device 'h14.private%9100' at 15:34:09.366
Status: printing job 'root@h4+223' at 15:34:09.375
Status: printing data file 'dfA223h4.private', size 3, IF filter 'ifhp' at 15:34:09.376
Status: IF filter finished at 15:34:35.012
Status: printing done 'root@h4+223' at 15:34:35.012
Status: accounting at end at 15:34:35.014
Status: finished 'root@h4+223', status 'JSUCC' at 15:34:35.014
Status: subserver pid 27251 exit status 'JSUCC' at 15:34:35.018
Status: lw4@h4.private: job 'root@h4+223' printed at 15:34:35.020
Status: job 'root@h4+223' removed at 15:34:35.101
Filter_status: accounting at start, pagecount 89696, pages 0 at 15:34:13.304
Filter_status: sending job file at 15:34:13.306
Filter_status: starting transfer at 15:34:13.306
Filter_status: initial job type 'text' at 15:34:13.306
Filter_status: job type 'pcl' at 15:34:13.306
Filter_status: transferring 3 bytes at 15:34:13.308
Filter_status: 100 percent done at 15:34:13.308
Filter_status: finished writing file, cleaning up at 15:34:13.308
Filter_status: sent job file at 15:34:13.308
Filter_status: doing cleanup at 15:34:13.308
Filter_status: getting end using 'pjl job/eoj' at 15:34:13.309
Filter_status: end of job detected at 15:34:33.219
Filter_status: getting pagecount using 'pjl info pagecount' at 15:34:33.219
Filter_status: final pagecount 89697 at 15:34:35.009
Filter_status: accounting at end, pagecount 89697, pages 1 at 15:34:35.010
Filter_status: done at 15:34:35.010
```

As you can see from the `lpq` output, the `Status` section shows the **lpd** activities and the Filter_status section shows what **ifhp** is doing. This is the first line of defense - make sure that the information and errors reported here are for the correct filter.

If this does not help, then we will start with the basic filter tests and work our way *back* to the print server. First, you will need a couple of test files. If you have a PostScript printer, find a simple PostScript file - the `ellipse.ps` file is included in the **ifhp** distribution in the UTILS directory. You will also need a short text file - `/etc/motd` is usually handy to use.

Create the `/tmp/send` file with the following contents:

```
#!/bin/sh
cp /dev/null /tmp/t
/usr/local/libexec/filters/ifhp -Tdev=/tmp/t,trace,debug=1 </etc/motd 2>/tmp/trace
```

This will create the `/tmp/t` file, run the **ifhp** filter, and put the trace and debugging information into the `/tmp/trace` file. Run this command using:

```
#!/bin/sh
sh -x /tmp/send
```

Examine the output in `/tmp/trace`. It will look like:

```
ifhp 15:46:14.402 [27307] main: Debug '1'
ifhp 15:46:14.403 [27307] main: dump <NULL>
ifhp 15:46:14.403 [27307] main: Model_id '<NULL>'
ifhp 15:46:14.440 [27307] main: scanning Raw for default, then model '<NULL>'
ifhp 15:46:14.441 [27307] Select_model_info: id 'default', list->count 1940, model-
>count 0, init 0
ifhp 15:46:14.448 [27307] Dump_line_list: main: Model information - count 156, max 204,
ifhp 15:46:14.448 [27307]   [ 0]='T=dev=/tmp/g,trace,debug=1'
ifhp 15:46:14.448 [27307]   [ 1]='banner@'
ifhp 15:46:14.448 [27307]   [ 2]='banner_file=/usr/local/libexec/filters/psbanner.ps'
ifhp 15:46:14.448 [27307]   [ 3]='converter='
ifhp 15:46:14.448 [27307]   [ 4]='debug=1'
ifhp 15:46:14.449 [27307]   [ 5]='default_language=text'
ifhp 15:46:14.449 [27307]   [ 6]='dev=/tmp/g'
ifhp 15:46:14.449 [27307]   [ 7]='duplex_select=1'
```

Most of the information with debug level 1 is simply showing the details of options, command execution, and error status. This will, however, help with the majority of problems.

You can now modify the `/tmp/send` file to better reflect your printer.

```
#!/bin/sh
cp /dev/null /tmp/t
# substitute your ifhp options here
ifhp=model=hp4,status@
/usr/local/libexec/filters/ifhp -Tdev=/tmp/t,trace,debug=1,${ifhp} </etc/motd 2>/tmp/tr
```

Now run this again and examine the trace and output in `/tmp/t`. If this looks correct, we move on to the interactive tests.

If your printer is a network based printer and you are using RFC1179 transfers, then you can use the following `lpr` command to send the `/tmp/t` file directly to the printer:

```
  lpr -Ppr@host /tmp/t
Example:
  print queue 'raw'

  lpr -Praw@10.1.1.1 /tmp/t
```

This form of the command causes `lpr` to send the job directly to the printer. If this works correctly then we know that there is no problem with **ifhp** formatting the file, and that the problem must be with the **lpd** print spooler.

If your printer is network based and uses a socket connection, then you can have **ifhp** connect to the printer by using the `dev=host%port` connection option. This is only used for testing or when you want to use the AppSocket protocol.

```
#!/bin/sh
# substitute your ifhp options here
#  this should always work
#ifhp=model=hp4,status@
#  this does status checking, no pagecount
#ifhp=model=hp4,pagecount@
#  this does status checking and pagecount
#ifhp=model=hp4
/usr/local/libexec/filters/ifhp -Tdev=10.1.1.1%9100,trace,debug=1,${ifhp} </etc/motd 2>
```

If you have a windowing system, run this command from one window and in another window use the `tail -f /tmp/trace` command to view status. Most of the time you will discover that the system is failing to print because either the `sync`, `pagecount`, or `waitidle` step of the printing process is not completing correctly. If there is insufficient detail for you to decide the failure mechanism, set `debug=3`, or even `debug=4` and explore what is happening.

If you have a parallel port connected printer on `/dev/lptxx`, then you can simply use `cat /tmp/t >/dev/lptxx` and see what happens. If this works, then use:

```
#!/bin/sh
# substitute your ifhp options here
ifhp=model=hp4,status@
/usr/local/libexec/filters/ifhp -Tdev=/dev/lptxxx,trace,debug=1,${ifhp} </etc/motd 2>/t
```

If you have a serial port printer, then you can use a similar method for setting up a connection. You will need to use the undocumented `stty` option to set the speed and other parameters. These are identical to those used by **LPRng**, so you should have no problems.

```
#!/bin/sh
# substitute your ifhp options here
#  this should always work
#ifhp=model=hp4,status@
#  this does status checking, no pagecount
#ifhp=model=hp4,pagecount@
#  this does status checking and pagecount
#ifhp="model=hp4,stty= 9600 -parity crtscts raw"
stty="9600 -parity crtscts raw"
/usr/local/libexec/filters/ifhp -Tdev=/dev/tty00,trace,debug=1,${ifhp},stty=\"${stty}\"
```

If all this does not help, then subscribe to the **LPRng** mailing list and ask for help.

Appendix A. Index to Options

Table A-1. `ifhp.conf` - ifhp Options

Option	Purpose
`default`	HP 4M Plus, PostScript, PJL, PCL, status, pagecount support
`appsocket FLAG`	Use Tektronix AppSocket Protocol
`config=PATHNAMES`	Configuration file pathnames
`crlf FLAG`	Do LF to CRLF translation
`debug FLAG`	Debugging level
`default_language=LANGUAGE`	Default job file language (ps, pcl, raw, text, etc)
`endpage=NNN`	PJL JOB command END = NNN value
`forceconversion FLAG`	Force conversion using UNIX file(1) utility
`file_output_match=LIST`	File type and conversion matching
`file_util_path=PATHNAME`	Pathname of the UNIX file(1) utility
`language=LANGUAGE`	Specify job file language to be used (ps, pcl, raw, text, etc)
`logall FLAG`	Log all status reports from printer if set
`nullpad=COUNT`	Send COUNT nulls to force full buffer condition
`model=NAME`	Specify model name for configuration selection
`model_from_option=X`	Specify model name using a command line option value
`pagecount=LANGUAGE`	Get pagecount using pjl, ps or default
`pagecount_interval=SECONDS`	Send pagecount command at SECONDS interval
`pagecount_ps_code=STRING`	PostScript code to get pagecount information
`pagecount_timeout=SECONDS`	Timeout getting pagecount after SECONDS
`pcl FLAG`	Printer supports PCL if set
`pcl_eoj_at_start FLAG`	PCL EOJ (CTRL-D) at start of job
`pcl_init=LIST`	PCL initializations to be done
`pcl_user_opts=LIST`	User PCL options supported
`pjl FLAG`	Printer supports PJL if set
`pjl_alert_codes=LIST`	alert operator on these PJL error codes

Option	Purpose
`pjl_alert_handler=STRING`	program to alert operator on PJL error
`pjl_console FLAG`	Printer supports messages on console
`pjl_error_codes=LIST`	PJL error messages for error codes
`pjl_except=LIST`	Do not allow these PJL commands
`pjl_init=LIST`	PJL initializations to be done
`pjl_job FLAG`	PJL JOB and EOJ supported
`pjl_only=LIST`	Allow only these PJL commands
`pjl_quiet_codes=LIST`	ignore these PJL error codes
`pjl_user_opts=LIST`	Allow only these user PJL commands or variables to be set
`pjl_vars_except=LIST`	Do not allow these PJL variables to be set
`pjl_vars_set=LIST`	Allow these PJL variables to be set
`ps FLAG`	Printer supports PostScript (ps)
`ps_eoj_at_start FLAG`	PostScript EOJ (CTRL-D) at start of job
`ps_init=LIST`	PS initializations to be done
`ps_user_opts=LIST`	Support these PostScript user options
`remove_ctrl=LIST`	Remove these characters from PostScript jobs
`remove_pjl_at_start FLAG`	Remove PJL commands from begining of job
`startpage=NNN`	PJL JOB command START = NNN value
`status FLAG`	Printer supplies status information
`statusfile=PATHNAME`	Status file pathname
`statusfile_max=NNN`	Status file has maximum size of NNN Kbytes
`statusfile_min=NNN`	Status file has truncated size of NNN Kbytes
`summaryfile=PATHNAME`	Summary file pathname
`sync FLAG`	Synchronize printer if set
`sync_interval=SECONDS`	Send synchronization request at SECONDS interval
`sync_timeout=SECONDS`	Timeout synchronization request after SECONDS
`tbcp FLAG`	Use Transparent Binary Communications Protocol for PostScript files
`text FLAG`	Printer supports text mode
`trace FLAG`	Put error and trace messages on STDERR if set
`waitend=METHOD`	How to wait for printer to end printing

Option	Purpose
`waitend_interval=SECONDS`	How often to query printer for end of printing
`waitend_ctrl_t_interval=SECONDS`	How often to send CTRL-T for end of printing status

Appendix B. HP JetDirect Card Support

The HPJetDirect card or external JetDirect box can be configured through the printer front panel or through a set of network files. Here is a summary of the methods used from UNIX systems, or when you are desperate, to configure the printer.

B.1. MicroSoft JetDirect Support

There is limited support from HP for non-MicroSoft configuration tools. However, the tools that are provided are simple and easy to use, especially for the initial configuration.

B.2. TCP/IP Address

You can set the TCP/IP address from the front panel. Reset the printer and then use the MENU, +-, SELECT keys as follows:

```
MENU  -> MIO MENU (use MENU to display MIO MENU)
ITEM  -> CFG NETWORK=NO*
+     -> CFG NETWORK=YES
ENTER -> CFG NETWORK=YES*
ITEM  -> TCP/IP=OFF* (use ITEM to display TCP/IP)
+     -> TCP/IP=ON
ENTER -> TCP/IP=ON*
ITEM  -> CFG TCP/IP=NO* (use ITEM to display TCP/IP)
+     -> CFG TCP/IP=YES
ENTER -> CFG TCP/IP=YES*
ITEM  -> BOOTP=NO*
 (Enable BOOTP if you want to - see below)
ITEM  -> IP BYTE 1=0*
 This is IP address MSB byte.
 Use +- keys to change value, and then ENTER to change
 Use ITEM keys to get IP BYTE=2,3,4
ITEM  -> SM BYTE 1=255*
  This is the subnet mask value
 Use +- keys to change value, and then ENTER to change
 Use ITEM keys to get IP BYTE=2,3,4
ITEM  -> LG BYTE 1=255*
 This is the Syslog server (LoGger) IP address
 Use +- keys to change value, and then ENTER to change
```

```
 Use ITEM keys to get IP BYTE=2,3,4
ITEM  -> GW BYTE 1=255*
 This is the subnet gateway (router) IP address
 Use +- keys to change value, and then ENTER to change
 Use ITEM keys to get IP BYTE=2,3,4
ITEM  -> TIMEOUT=90
  This is the connection timeout value.  It puts a limit
 on time between connections.  A value of 10 is reasonable.
```

B.3. Web Server Configuration

Many of the newer releases of HP JetDirect firmware have a Web Server configuration capability. Unfortunately, the web pages assume that you have JavaScript support for the browser, and not all of the facilities used are supported by all browsers. However the configuration information and pages presented are extremely simple to use and understand.

The major problem with the Web Server configuration is that not all of the options can be set through the Web Server pages.

B.4. Telnet Configuration

Once you have assigned an IP address to the JetDirect Box you can telnet to the box and configure it through a simple command line interface. When you first connect to the box, hit `RETURN` a couple of times, followed by `?` (Question Mark) `RETURN` and you should get a simple help menu. This help information usually scrolls off a small (25 line) screen so a buffer that allows you to see all of the information is advisable.

The major problem with the telnet configuration is that not all of the options can be set through the command line facilties.

B.5. BOOTP Information

If you have a bootp server, you can put this information in the bootptab file. To use this, you must enable the bootp option on the printer. The T144 option specifies a file to be read from the bootp server. This file

is read by using the TFTP protocol, and you must have a TFTPD server enabled. Here is a sample bootptab entry.

```
# Example /etc/bootptab: database for bootp server (/etc/bootpd).
# Blank lines and lines beginning with '#' are ignored.
#
# Legend:
#
#       first field -- hostname
#                         (may be full domain name)
#
#       hd -- home directory
#       bf -- bootfile
#       cs -- cookie servers
#       ds -- domain name servers
#       gw -- gateways
#       ha -- hardware address
#       ht -- hardware type
#       im -- impress servers
#       ip -- host IP address
#       lg -- log servers
#       lp -- LPR servers
#       ns -- IEN-116 name servers
#       rl -- resource location protocol servers
#       sm -- subnet mask
#       tc -- template host (points to similar host entry)
#       to -- time offset (seconds)
#       ts -- time servers
#
# Be careful about including backslashes where they're needed.  Weird (bad)
# things can happen when a backslash is omitted where one is intended.
#
peripheral1:
:hn:ht=ether:vm=rfc1048:
:ha=08000903212F:
:ip=190.40.101.22:
:sm=255.255.255.0:
:gw=190.40.101.1:
:lg=190.40.101.3:
:T144="hpnp/peripheral1.cfg":
```

If you are using the T144 option, you will need to create the configuration file. The sample configuration file from the HP Direct distribution is included below.

```
#
# Example HP Network Peripheral Interface configuration file
#
# Comments begin with '#' and end at the end of the line.
# Blank lines are ignored.  Entries cannot span lines.

# Name is the peripheral (or node) name.  It is displayed on the peripheral's
# self-test page or configuration plot, and when sysName is obtained through
# SNMP.  This name can be provided in the BOOTP response or can be specified
# in the NPI configuration file to prevent the BOOTP response from overflowing
# the packet.  The domain portion of the name is not necessary because the
# peripheral does not perform Domain Name System (DNS) searches.  Name is
# limited to 64 characters.

name: picasso

# Location describes the physical location of the peripheral.  This is the
# value used by the interface for the MIB-II sysLocation object.  The default
# location is undefined.  Only printable ASCII characters are allowed.
# Maximum length is 64 characters.

location: 1st floor, south wall

# Contact is the name of the person who administers or services the peripheral
# and may include how to contact this person.  It is limited to 64 characters.
# This is the value used by the interface for the MIB-II sysContact object.
# The default contact is undefined.  Only printable ASCII characters are
# allowed.  Maximum length is 64 characters.

contact: Phil, ext 1234

# The host access list contains the list of hosts or networks of hosts
# that are allowed to connect to the peripheral.  The format is
# "allow: netnum [mask]", where netnum is a network number or a host IP
# address.  Mask is an address mask of bits to apply to the network number
# and connecting host's IP address to verify access to the peripheral.
# The mask usually matches the network or subnet mask, but this is not
# required.  If netnum is a host IP address, the mask 255.255.255.255 can
# be omitted.  Up to ten access list entries are permitted.

# to allow all of network 10 to access the peripheral:
```

```
allow: 10.0.0.0  255.0.0.0

# to allow a single host without specifying the mask:
allow: 15.1.2.3

# Idle timeout is the time (in seconds) after which an idle
# print data connection is closed.  A value of zero disables
# the timeout mechanism.  The default timeout is 90 seconds.

idle-timeout: 120

# A community name is a password that allows SNMP access to MIB values on
# the network peripheral.  Community names are not highly secure; they are
# not encrypted across the network.  The get community name determines which
# SNMP GetRequests are responded to.  By default, the network peripheral
# responds to all GetRequests.  The get community name is limited to 32
# characters.
#
# For hpnpstat and hpnpadmin, the community name can be stored in
# /usr/lib/hpnp/hpnpsnmp.

get-community-name: blue

# The set community name is similar to the get community name.  The set
# community name determines which SNMP SetRequests are responded to.  In
# addition, SetRequests are only honored if the sending host is on the
# host access list.  By default, the network peripheral does not respond
# to any SetRequests.  The set community name is limited to 32 characters.
#
# The set community name can come from /usr/lib/hpnp/hpnpsnmp
# if it is the same as the get community name.  We recommend that the
# set community name be different from the get community name though.

set-community-name: yellow

# SNMP traps are asynchronous notifications of some event that has occurred.
# SNMP traps are useful only with network management software.  Traps are
# sent to specific hosts and include a trap community name.  Up to four
# hosts can be sent SNMP traps.   The trap community name is limited to
# 32 characters.  The default name is public.

trap-community-name: red

# The SNMP trap destination list specifies systems to which SNMP
# traps are sent.  Up to four IP addresses are allowed.  If no
```

```
# trap destinations are listed, traps are not sent.

trap-dest: 15.1.2.3
trap-dest: 15.2.3.4

# The SNMP authentication trap parameter enables or disables the sending
# of SNMP authentication traps.  Authentication traps indicate that an SNMP
# request was received and the community name check failed.  By default,
# the parameter is off.

authentication-trap: on

# The syslog-facility parameter sets the source facility identifier that the
# card uses when issuing syslog messages.  Other facilities, for example,
# include the kernel (LOG_KERN), the mail system (LOG_MAIL), and the spooling
# system (LOG_LPR).  The card only allows its syslog facility to be configured
# to one of the local user values (LOG_LOCAL0 through LOG_LOCAL7).  The
# selectable option strings, local0 through local7 (configured to LOG_LOCAL0
# through LOG_LOCAL7, respectively) are case insensitive.  The default
# syslog-facility for the card is LOG_LPR.

syslog-facility: local2

# This parameter allows the card to treat hosts on other subnets as if the
# hosts were on the card's subnet.  This parameter determines the TCP
# Maximum Segment Size (MSS) advertised by the card to hosts on other subnets
# and affects the card's initial receive-window size.  The card will use a
# TCP MSS of 1460 bytes for local hosts, and 536 bytes for a non-local host.
# The default is off, that is, the card will use the maximum packet sizes
# only on the card's configured subnet.
#
# The configuration utility does not allow access to this parameter.  If you
# want to configure it, you must manually edit the NPI configuration file
# and add it to the bottom of the entry for the network peripheral.

subnets-local: on

# This parameter affects how the card handles TCP connection requests from
# the host.  By default, the JetDirect MPS card will accept a TCP connection
# even if the peripheral is off-line.  If this parameter is set to "on", then
# the card will only accept a TCP connection when the peripheral is on-line.

old-idle-mode: off
```

B.6. Timeouts

You should be aware that the `idle-timeout` value in the configuration file will override the value entered on the control panel of the printer.

Also, the `@PJL SET TIMEOUT = NNN` command will override this value as well.