IFHP-HOWTO

Patrick Powell *papowell@astart.com* 17 May 1999 (For ifhp-3.2.9)

The `ifhp` software is an enhanced, extended, highly configurable, and portable implementation of a print filter for use with the `ifhp` Print spooler software. It supports network, serial, and parallel printers, does page accounting and job recovery, and allows an extremely high level of configuration and tuning. The `ifhp` filter gets its flexibility by using a configuration file to set its operational characteristics. The configuration file can contain multiple separate printer configurations, and the desired one can be selected by a simple mechanism. The filter can support vintage text, PostScript, PCL, and PJL printers, and can be configured to handle a wide range of printer quirks and misimplementations.

Contents

1 Introduction

The `ifhp` print filter is the latest in a long evolutionary path of print filters for the `ifhp` printing system. It is intended to unify several different methods of controlling printers, and provide a common code base for future development.

This document is the complete set of references and installation guide for the `ifhp` printer. It covers the compilation and installation, initial testing, details of system configuration, and various configuration options that would be needed by the system administrator.

The reference for Printer Job Language (PJL) related issues was the Printer Job Language Technical Reference Manual, Hewlett Packard, 10th Edition, October 1997, and the reference for PCL related issues was 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.

Previous releases of `ifhp` had a large selection of `README` files which are now incorporated into the `ifhp`-HOWTO document.

Current information about LPRng, `ifhp` and the latest release can be found on the LPRng web page:

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

There is also a mailing list at `lprng@iona.com`. To subscribe, send an email to `lprng-request@iona.com`. The body should contain only the word 'subscribe'. To get off the list later on, repeat the same procedure, but use the word 'unsubscribe'.

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. *<tt>ifhp</tt> - An Enhanced Printer Spooler System* was presented at the LISA95 conference, and is in the LPRng distribution as LPRng-LISA95.ps. On a more general topic, the slides for the LISA97 tutorial on *Printers and Network Print Spooling* are also in the LPRng distribution in the DOC/LISA97 directory.

1.1 Copyright

Material included in this document from the `ifhp` distribution Copyright Patrick Powell 1988-1999, where applicable.

The rights to distribute this document complete or in part are hereby granted for non-commercial purposes. Partial reproductions must acknowledge the source.

Permission to distribute this file together with LPRng and 'derived works' (as defined in the LPRng license) is explicitly granted. This is allowed independent of the license under which the software is distributed.

Citing the document is allowed as long as the source is acknowledged.

1.2 Disclaimer

THE MATERIAL IN THIS HOWTO IS PROVIDED WITHOUT FEE AND AS-IS WITH NO WARRANTY REGARDING FITNESS OF USE FOR ANY PURPOSE. THE AUTHOR AND ALL CONTRIBUTORS ARE NOT LIABLE FOR ANY DAMAGES, DIRECT OR INDIRECT, RESULTING FROM THE USE OF INFORMATION PROVIDED IN THIS DOCUMENT.

1.3 Commercial Support

AStArt Technologies 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.

1.4 Web Site

Web Page:

http://www.astart.com/lprng.html

1.5 FTP Sites

The software may be obtained from
ftp://ftp.astart.com/pub/LPRng/FILTERS(Main site)

Mirrors:
ftp://ftp.sage-au.org.au/pub/printing/spooler/lprng (AU)
ftp://ftp.zod.wau.nl/pub/mirror/plp/LPRng (AU/NZ)
ftp://gwynne.cs.ualberta.ca/pub/LPRng (CA)
ftp://ftp.informatik.uni-hamburg.de/pub/os/unix/utils/LPRng (DE)
ftp://ftp.uni-paderborn.de/pub/unix/printer/plp/LPRng (DE)
ftp://ftp.iona.ie/pub/plp/LPRng (IE)
ftp://ftp.chembio.ntnu.no/pub/mirrors/LPRng (NO)
ftp://ftp.mono.org/pub/LPRng (UK)
ftp://ftp.cs.columbia.edu/pub/archives/pkg/LPRng (US)
ftp://ftp.cs.umn.edu/pub/LPRng (US)
ftp://ftp.iona.com/pub/plp/LPRng (US)
ftp://uiarchive.uiuc.edu/pub/packages/LPRng (US)

1.6 Mailing List

To join the LPRng mailing list, please send mail to *lprng-request@iona.ie* with the word 'subscribe' in the BODY

1.7 PGP Public Key

The LPRng and `ifhp` distributions have an MD5 checksum calculated, which is then 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+0Kzx3/y/0aK4cizoWqKvy1l4lEzDsA2VydAAUT
tCdQYXRyaWNrIEEuIFBvd2VsbCA8cGFwb3dlbGxAYXN0YXJ0LmNvbT6JAJUDBRA0
XonoiUTMOwDZXJ0BAQ2cBAC7zU9Fn3sC3x0USJ+3vjhg/qA+Gjb5Fi1dJd4solc4
vJvtf0UL/1/rGipbR+A0XHpHzJUMP9ZfJzKZjaK/d0ZBNlS3i+JnypypeQiAqo9t
FV0OyUCwDfWybgA0RuAa2V6UJnAhvj/7TpxMmCApolaIb4yFyKunHa8aBxN+17Ro
rrQlUGF0cmljayBBLiBQb3dlbGwgPHBhcG93ZWxsQHNkc3UuZWR1PokAlQMFEDLy
```

```
gTSJRMw7ANlcnQEBYBYD/0zTeoiDNnI+NjaIei6+6z6oakq070qFVx0FG3aP3kRH
WlDhdtFaAuaMRh+RItHfFfcHhw5K7jiJdgKiTgGfj5Vt30dHYkeeh/sddqgf9YnS
tpj0u5NfrotPTUw39n6YTgS5/aW0PQf09dx7jVUcGeod1TGXTe9mIhDMwDJI4J14
=3Zbp
-----END PGP PUBLIC KEY BLOCK-----
```


2 Installation and Express Configuration

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

1. Use GNU Make. Don't even think about trying to use another make unless you are a Wizard. And even the Wizards would download the GNU Make.
2. Use an ANSI C compiler.
3. Read the `HOWTO/ifhp-HOWTO.{ps,html,text,...}` file(s). You are doing this now, so you are off to a good start.
4. Generate the executables.
5. Install the configuration files.
6. Run some simple standalone tests, i.e. - without using the print spooler.
7. Modify the configuration files to suit your sites requirements.
8. Install the executables.
9. Modify the printcap files and/or add other information to the printcap files.
10. Try the `ifhp` filter with the working print spooler.

In addition, you might want to get the following software, which can be used with `ifhp`.

a2ps - Ascii Text To PostScript Converter

http://www-inf.enst.fr/~demaille/a2ps/ This package does a very good job of text to PostScript conversion.

enscript - GNU Enscript

http://www.gnu.org/ This package is a simpler version of a2ps, and is faster and smaller.

Unix File Utility - Determines the type of file

ftp://ftp.astron.com/pub/file/ or *ftp://ftp.deshaw.com/pub/file/*.

LPRng Print Spooler

http://www.astart.com

2.1 Configure and Compilation

The `ifhp` filter uses the AUTOCONF configuration facilty. The following set of commands will generate configuration files and compile and install the software and documentation.

It is highly unlikely that you will encounter problems with compilation. Usually these are due to type definition conflicts in include files. If you encounter these, please report these to the LPRng mailing list.

```
configure
make all
# installs ifhp and textps in /usr/local/lib/filters
# installs ifhp.conf in /etc
#  (if already present, in /etc/ifhp.conf.sample)
make install
```

2.2 Printcap Configuration

The basic LPRng printcap configuration is:

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

# version 1 - using -T options
#path to ifhp filter
:if=/.../ifhp -Toptions
:of=/.../ifhp -Toptions

# version 2 - using ifhp printcap entry
:ifhp=options
:if=/.../ifhp
:of=/.../ifhp
```

As shown, options can be passed on the command line using the `-Toption,option` method or put into the `:ifhp=option,option` list.

Since commas are used to separate options in the `-T` option list, if you want to specify an option with a list of values the values need to separated with a semicolon (`;`) as shown below:

```
ifhp -Tconfig=./ifhp.conf;/etc/ifhp.conf;./ifhp.conf
```

By convention, the `ifhp` filter takes its input from STDIN (file descriptor 0) and expects to write its output to STDOUT (file descriptor 1), which is normally connected to a printer. Error and trace information are written to STDERR (file descriptor 2).

As shipped, the `ifhp.conf` file is configured to support a PJL printer which has the following configuration. The configuration flags that support or enable these features are shown as well.

1. PJL support (`pjl`).
2. PostScript (PS) support (`ps`).
3. PCL support (`pcl`).
4. Text files printed as PCL (`text`, `default_language=pcl`).
5. Banner printing done by LPRng spooler (`banner@`)
6. Status reported from printer (over bidirectional file descriptor 1) (`status`)

7. Synchronize at start and end (`sync`)
8. Get pagecount information (`pagecount`)

The following printers have specfic configuration sections which are invoked by using the `model=name` option. The HP DeskJet and DesignJet printers use the same `hpdj` prefix with the model appended. See Appendix A in the Printer Job Language Technical Reference Manual for a detailed explanation of these names.

```
apple      hp4000     hp4         hp4l        hp4lc
hp4m       hp4ml      hp4mp       hp4mplus    hp4mv
hp4p       hp4pj      hp4plus     hp4si       hp4simx
hp4v       hp5        hp5l        hp5m        hp5mp
hp5p       hp5si      hp5simopier hp5simx     hp6l
hp6mp      hp6p       hpcolorlj5  hpcolorlj5m hpcolorlj
hpdj1200c  hpdj1600c  hpdj2000cp  hpdj200     hpdj220
hpdj230    hpdj2500cp hpdj250c    hpdj330     hpdj350c
hpdj350c   hpdj430    hpdj450c    hpdj455ca   hpdj600
hpdj650c   hpdj700    hpdj750c    hpdj750cplus hpdj755cm
hpiiisi    hpljpro    hppjxl300   postscript  ps
tek        qms1725
```

In addition to HP printers, there is also generic support for PostScript only printers.

2.3 PS, PCL, PJL Printer with Network Connection

This is the most common configuration, and the printcap entry would have the following format:

```
# printer setup
# force clients (lpr, lpq, to use server)
lp:lp=lp@serverhost
# server information
lp:server
:sd=spooldir
:...
#:lp=printer ip address or DNS name%9100
# eg - :lp=nwpr%9100
# eg - :lp=10.1.1.1%9100
:lp=10.1.1.1%9100
#path to ifhp filter
:if=/.../ifhp
:of=/.../ifhp
```

If you have an HP printer in the above list, you can use:

```
# printer setup
# force clients (lpr, lpq, to use server)
lp:lp=lp@serverhost
# server information
lp:server
:sd=spooldir
:...
:lp=printer ip address or DNS name%9100
```

```
:ifhp=model=name   #path to ifhp filter
:if=/.../ifhp
:of=/.../ifhp
```

2.4 PS, PCL, PJL Printer with Parallel Port Connection

Since the parallel port is unidirectional, you cannot get status back, and need to use the `status@` option to prevent the `ifhp` filter from expecting it. The printcap entry would have the following format:

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

2.5 PS, PCL, PJL Printer with Serial Port

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. for your system,

```
# 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
:if=/.../ifhp
:of=/.../ifhp
```

2.6 PostScript Only Printer

Use the configuration appropriate to the printer connection, and then use the `pjl@`, `pcl@`, and `text@` option to inhibit everything but PostScript.

```
# printer setup
# force clients (lpr, lpq, to use server)
lp:lp=lp@serverhost
# server information
lp:server
:sd=spooldir
:...
:ifhp=pcl@,pjl@,text@
#path to ifhp filter
:if=/.../ifhp
:of=/.../ifhp
```

If you have a parallel port printer with no PostScript support, you would use:

```
:ifhp=pcl@,pjl@,text@
```

An alternative is to use the `model=ps` configuration section of the default `/etc/ifhp.conf` file.

```
# 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
:if=/.../ifhp
:of=/.../ifhp
```

If your printer does not like to have a PostScript EOJ (Control-D) flag at the start of a job, use the `no_ps_eoj` (3.11 (No PS EOJ at Start)) flags. flag to remove them.

2.7 PostScript Only Printer With Text Conversion

You can use the PostScript Only configuration above, and add text to postscript conversion to it. This is done by defining the pathname to a text to PostScript conversion program and causing ifhp to use the program to convert the input file to PostScript. The *a2ps* program can be obtained from *http://www-inf.enst.fr/~demaille/a2ps/* and does an excellent job. Also, the `textps` program included with this distribution does a (barely) adequate job as well.

The method outlined here assumes that the input file is a text file and that the `a2ps` program will convert it. The `a2ps` program needs to be explicitly told that output is to go to `STDOUT` or file descriptor 1.

```
text_converter=/usr/local/bin/a2ps -o-
text_converter_output=ps
tempfile=/var/tmp/ifhp

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

```
lp:server
:sd=spooldir
:...
:ifhp=pcl@,pjl@,text@,text_converter=...,\
text_converter_output=ps
#path to ifhp filter
:if=/.../ifhp
:of=/.../ifhp
```

2.8 Tektronixs Phaser Series Printer with Network Connections

The Tektronics Phaser Series printers use the *Appsocket* protocol when sending a job to the printer. This protocol is (briefly):

1. The printer listens for UDP packets on port 9101 and for TCP/IP connections on port 9100.
2. When a UDP packet is recieved on port 9101, then a reply packet containing the status is returned to the originator's address. This packet contains an status indication, in a *undefined* format.
3. To send a job to the printer, a TCP/IP connection is opened to port 9100, and a PostScript job is sent. Only a single job can be sent at a time - a EOJ (CTRL-D) will terminate input and flush all following jobs.
4. Return status will be sent in the reverse direction until the job has completed, at which point the connection will be closed.

The `ifhp` program supports the `appsocket` protocol. In order to do so, the following ifhp configuration entry can be used.

```
[ tek ]
appsocket
pjl@
pcl@
```

The `appsocket` option will use the appsocket protcol, and cause `ifhp` to open and close connections to the printer. The printcap entry should specify `lp=/dev/null` and provide the device IP address using the `-Tdev=host%port` option.

```
# Phaser Setup lp:server
:lp=/dev/null
:sd=spooldir
:...
:ifhp=model=tek,dev=10.0.0.1%9100
#path to ifhp filter
:if=/.../ifhp
:of=/.../ifhp
```

3 Configuration Tutorial

This section is a basic tutorial on using many of the features of the `ifhp` filter. If you plan to make use of any of the advanced filter options or modify the configuration, you should read and try out several of the configuration exercises.

3.1 Configuration Files and Testsupport Directory

These exercises require the use of several configuration files that have predefined values as well as the default /etc/ifhp.conf file. The test files are normally located in the distribution testscripts directory. It is assumed that these exercises will be done in that directory.

The config=path;path;path option specifies the list of configuration files for ifhp to read. Here is a sample of the configuration file t1.conf:

```
# modify the next line to contain
#  the name, IP address, or pathname of the printer
dev=CHANGE_THIS
debug=3
stty=38400 -echo -crmod -raw -oddp -evenp ixon pass8 -ixany cbreak crtscts
trace
ps@
pjl@
pcl@
text
```

The command

```
ifhp -T/etc/ifhp.conf;t1.conf
```

will cause the /etc/ifhp.conf and then the t1.conf file to be read and configuration information extracted.

The various options in this file are quite important for debugging and testing. The debug and trace options can also be specifed on the command line, and cause the debug facilities to be turned on and debug trace output to be put to STDERR. Normally, trace output is written only to the statusfile specified by the command line -s option or the configuration file statusfile=path entry.

The text entry causes the filter to accept text files and simply pass them through without processing, and the ps@ pjl@ pcl@ entries disable related processing.

3.2 Output Device Specification

In normal operation, ifhp sends its output to STDOUT or file desciptor 2. However, when testing or using the ifhp filter in standalone mode, it is useful to have the filter open the connection to a file or device directly. The following sections discuss how to modify the configuration file to do this, as well as how to implement a simple a

3.3 Output to a File

This method is useful when trying to debug a set of configuration options and you want to capture output to a file. The dev=path option is used to specify the output pathname.

1. In the testscripts directory, copy the t1.conf file to ifhp.conf and edit ifhp.conf so it has the following lines:

```
# dev=/tmp/out
# debug=3
stty=38400 -echo -crmod -raw -oddp -evenp ixon pass8 -ixany cbreak crtscts
trace
ps@
```

```
pjl@
pcl@
text
```

2. Run the indicated commands and you should see similar output:

```
##> ifhp -Tconfig=./ifhp.conf < hi.txt
ifhp 12:28:20.603 [6089] Process_job: setting up printer
ifhp 12:28:20.605 [6089] Do_accounting: accounting at start, pagecount 0, pages 0
ifhp 12:28:20.605 [6089] Process_job: sending job file
ifhp 12:28:20.605 [6089] Send_job: starting transfer
Hello World

ifhp 12:28:20.606 [6089] Send_job: finished writing file, cleaning up
ifhp 12:28:20.606 [6089] Process_job: sent job file
ifhp 12:28:20.606 [6089] Process_job: doing cleanup
ifhp 12:28:20.606 [6089] Do_accounting: accounting at end, pagecount 0, pages 0
ifhp 12:28:20.606 [6089] Process_job: done
```

3. Now modify the `ifhp.conf` file to look like:

```
dev=/tmp/out
debug=3
trace
ps@
pjl@
pcl@
text
```

4. Run indicated commands and you should see similar output:

```
##> ifhp -Tconfig=./ifhp.conf < hi.txt
ifhp 12:33:39.504 [6155] Open_device: device '/tmp/out'
ifhp 12:33:39.505 [6155] Open_device: device '/tmp/out', attempt 1
ifhp 12:33:39.506 [6155] Open_device: success
ifhp 12:33:39.506 [6155] main: fd 1 device 0100000
 ...
```

5. Clearly this is very verbose diagnostic or trace output. However, if you use `cat /tmp/out` you will see the output has been written correctly. Try using the following command to redirect trace and error output to a file for later examination.

```
CSH                      SH
ifhp ... <in >&out       ifhp ... <in 2>out
```

3.4 Default ifhp.conf File and Simple CRLF Filter

In order to simplify operation, if there is a `ifhp.conf` file in the current directory, then the `ifhp` program will read this file first, followed by `/etc/ifhp.conf` and then rereads the `ifhp.conf` This has the effect of extracting critical option information on the first pass through `ifhp.conf`, then reading the default information from `/etc/ifhp.conf`, and finally re-reading `ifhp.conf` to override any defaults.

1. Try the simple commands

```
echo >/tmp/out
ifhp <hi.txt
```

and note that it produces the same output as befoe.

2. Now modify the `ifhp.conf` file to contain:

```
dev=/tmp/out
debug=3
trace
ps@
pjl@
pcl@
text
crlf
```

and do

```
ifhp <hi.txt
```

again, without clearing the output file.

3. If you examine `/tmp/out`output using the `od` or `hexdump` file dump programs or the `vi` text editor, you will find that the LF (`\n`) character has been replaced by CR LF (`\r\n`).

4. You can use the `ifhp` program as a simple CRLF filter by specifying:

```
ifhp -Tcrlf,language=unknown <hi.txt
```

The `language=unknown` option specifies that the ifhp program is not to guess at the language type, and then does CRLF translation.

3.5 HP Jetdirect or Compatible Network Interface

If a printer has HP Jetdirect compatibility, then the printers TCP/IP port 9100 will accept jobs for printing and provide printer status.

Other manufacturers make similar products, and may support equivalent functionality on different ports. If you are using a Network Printer Box with multiple printer ports, then each printer port is accessible using a specific TCP/IP port. For example, parallel ports P1, P2, and P3 and on a JetDirect box are usually accessed using TCP/IP ports 9100, 9101, and 9102 respectively.

If you are using a Network Printer Box and are connecting to a parallel port printer, then you may not get status or error messages back from the printer unless both the printer and the Network Printer Box use bi-directional communication.

In the following discussions, we will assume that you have assigned an IP address to the printer, and that it is `nwpr`, and that status and error information is returned over the network connection.

1. First, verify that your printer is accessible on the network by using `ping`. It is amazing how many problems are caused by erroneous network addresses.

```
astart:  % ping nwpr
PING nwpr.astart.com (10.0.0.10):  56 data bytes
64 bytes from 10.0.0.10:  icmp_seq=0 ttl=60 time=4.286 ms
```

2. Now use `telnet` to check that you can connect to port 9100.

```
astart: % telnet nwpr 9100
Trying 206.71.174.206...
Connected to nwpr.astart.com.
Escape character is '^]'.
...
```

3. If you have a PostScript printer, then you might want to play with the PostScript Interactive mode, described in Appendix D.5 of the PostScript Language Reference Manual by Adobe Systems. Simply type the command

   ```
   executive
   ```

   as the first characters that are sent. You should see:

   ```
   >># telnet nwpr
   Connected to nwpr.astart.com.
   Escape character is '^]'.
   executive
   PostScript(r) Version 2013.111
   (c) Copyright 1984-1993 Adobe Systems Incorporated.
   Typefaces (c) Copyright 1981 Linotype-Hell AG and/or its subsidiaries.
   All Rights Reserved.
   PS>
   ```

4. You may want to try sending a simple file to the printer and examining what happens. This is easily done using the `netcat` program by Mudge. See *http://www.l0pht.com/~weld/netcat/index.html* for general information.

5. The `hi.txt` file contains

   ```
   Hello World
   ^L
   ```

   where `^L` is the form feed character You can send this file to the printer using:

   ```
   nc printer 9100 <hi.txt
   ```

   The printer should start up and print a single page of output. You may need to use `CTRL-C` to terminate the `nc` program.

6. If you have a PostScript printer, you might try sending a simple PostScript test program to the printer as well. The `ellipse.ps` file can be sent using:

   ```
   nc printer 9100 </tmp/hi1.txt
   ```

7. You can now test the basic `ifhp` functionality.

8. Copy the `t1.conf` file to `ifhp.conf` and edit is so it has the correct network printer information.

   ```
   # modify the next line to contain
   # the name, IP address, or pathname of the printer dev=nwpr%9100
   debug=3
   trace_to_stderr
   ps@ pjl@ pcl@ text
   ```

9. Now do the indicated command; you should get similar output

   ```
   astart: % ifhp <hi.txt
   ```

   Note the extreme verbosity of output.

10. At this point you might want to try getting page numbers and doing synchronization. If you have a HP printer that supports PJL, then modify `t2hp.conf` as above. This file has the contents:

```
# modify the next line to contain <newline>
#  the name, IP address, or pathname of the printer
dev=<it/nwpr/%9100 <newline>
#debug=3 <newline>
trace_to_stderr <newline>
text <newline>
pjl <newline>
sync=pjl <newline>
pagecount=pjl <newline>
```

Now use the following command and you should see similar output.

```
ifhp '-Tconfig=t2hp.conf;/etc/ifhp.conf;t2hp.conf' < hi1.txt
ifhp 16:44:16.541 [3199] Process_job: setting up printer
ifhp 16:44:16.545 [3199] Do_sync: starting sync using 'pjl'
ifhp 16:44:17.935 [3199] Check_device_status: device = 'Toner Low'
ifhp 16:44:18.116 [3199] Do_sync: sync done
ifhp 16:44:18.117 [3199] Do_pagecount: getting pagecount using 'pjl'
ifhp 16:44:20.011 [3199] Do_pagecount: final pagecount 63304
ifhp 16:44:20.013 [3199] Do_accounting: accounting at start, pagecount 63304, pages 0
ifhp 16:44:20.013 [3199] Process_job: sending job file
ifhp 16:44:20.014 [3199] Send_job: starting transfer
ifhp 16:44:20.115 [3199] Use_file_util: file identified as 'ascii text ', assigned type 'text'
ifhp 16:44:20.119 [3199] Send_job: finished writing file, cleaning up
ifhp 16:44:20.119 [3199] Process_job: sent job file
ifhp 16:44:20.119 [3199] Process_job: doing cleanup
ifhp 16:44:20.120 [3199] Do_sync: starting sync using 'pjl'
ifhp 16:44:22.175 [3199] Do_sync: sync done
ifhp 16:44:22.176 [3199] Do_waitend: getting end using 'pjl'
ifhp 16:44:22.529 [3199] Check_device_status: device = 'Warming Up'
ifhp 16:45:04.669 [3199] Check_device_status: device = 'Toner Low'
ifhp 16:45:21.361 [3199] Do_pagecount: getting pagecount using 'pjl'
ifhp 16:45:23.241 [3199] Do_pagecount: final pagecount 63305
ifhp 16:45:23.243 [3199] Do_accounting: accounting at end, pagecount 63305, pages 1
ifhp 16:45:23.243 [3199] Process_job: done
```

11. At this point, you should be able to send jobs to the printer, and now need to put these options into the printcap file. Please see the section on 3.8 (Printcaps and Fine Tuning Configurations)

3.6 Serial Port

If your printer is attached using a serial port, then you must make sure that the communication channel is configured correctly. The `dev=/dev/ttyxxx` and `stty=...` configuration options will open the serial port and configure the line in an appropriate manner. For details on the exact form of the `stty` options, please consult the LPRng-HOWTO documentation.

1. Copy the `t1.conf` file to `ifhp.conf` and edit it so it has the following lines:

```
# modify the next line to contain
# the name, IP address, or pathname of the printer dev=/dev/serialport
#debug=3
stty=38400 -echo -crmod -raw -oddp -evenp ixon pass8 -ixany cbreak crtscts
trace_to_stderr
text
```

Now run the `ifhp` program as shown below. You should see output similar to:

```
astart: % ifhp <h11.txt
ifhp 15:57:23.734 [3096] Process_job: setting up printer
ifhp 15:57:23.736 [3096] Do_accounting: accounting at start, pagecount 0, pages 0
ifhp 15:57:23.736 [3096] Process_job: sending job file
ifhp 15:57:23.736 [3096] Send_job: starting transfer
ifhp 15:57:23.737 [3096] Send_job: finished writing file, cleaning up
ifhp 15:57:23.737 [3096] Process_job: sent job file
ifhp 15:57:23.738 [3096] Process_job: doing cleanup
ifhp 15:57:23.738 [3096] Do_accounting: accounting at end, pagecount 0, pages 0
ifhp 15:57:23.738 [3096] Process_job: done
```

and you should get a printed page.

2. If you do not get communication, then you will have to start a long and painful process of debugging. First, check that the various modem control signals are correct. You should invest a small amount of time and energy and purchase a **breakout box** that shows the status of the various control signals. If only the **XMIT Data** signal is active, you will need to put a crossover (null modem) between the printer and your serial port. Check the **DTR** (Data Terminal Ready), **DCD** (Data Carrier Detect), **RTS** (Request To Send), and **CTS** (Clear To Send). These should all be ON or active. Some systems will use **RTS** (Request To Send) and **CTS** (Clear To Send) for *hardware flow control* and it is possible that these signals will become stuck. It is strongly recommended that flow control on the host be turned off, and let the printer do flow control using **RTS** and **CTS**.
3. Once you get the basic configuration working, then you should follow the basic steps in the previous section on Network Interfaces, and try sending files of different types.

3.7 Parallel Port

The parallel port configuration is the easiest to implement, but the most limited in abilities.

1. Copy the `t1.conf` file to `ifhp.conf` and edit it so it has the following lines, with the `dev` value replaced by the correct pathname.

```
# modify the next line to contain
# the name, IP address, or pathname of the printer
dev=/dev/lp
debug=3
stty=38400 -echo -crmod -raw -oddp -evenp ixon pass8 -ixany cbreak crtscts
trace
ps@
pjl@
pcl@
text
```

2. Now run the `ifhp` program:

```
ifhp <hi.txt
```

3. You should see output similar to:

```
astart: % ifhp -Tconfig=.../testscripts/t1.conf <h11.txt
ifhp 15:57:23.734 [3096] Process_job: setting up printer
```

```
ifhp 15:57:23.736 [3096] Do_accounting: accounting at start, pagecount 0, pages 0
ifhp 15:57:23.736 [3096] Process_job: sending job file
ifhp 15:57:23.736 [3096] Send_job: starting transfer
ifhp 15:57:23.737 [3096] Send_job: finished writing file, cleaning up
ifhp 15:57:23.737 [3096] Process_job: sent job file
ifhp 15:57:23.738 [3096] Process_job: doing cleanup
ifhp 15:57:23.738 [3096] Do_accounting: accounting at end, pagecount 0, pages 0
ifhp 15:57:23.738 [3096] Process_job: done
```

and you should get a printed page. If you happen to have an operating system that support bi-directional parallel ports, you can add to the ifhp

```
read_write
```

which will cause ifhp to open the port read/write.

3.8 Fine Tuning A Configuration and Printcap Entries

If you have followed the above directions, you should now be able to use the IFHP filter with very little additional configuration. You can now set up your printcap to use the `ifhp` filter. The following discussion assumes that you are using LPRng.

3.9 ifhp.conf Defaults

While the above set of exercises are useful for debugging and testing, you should always let the LPRng spooler open the printing device or make the network connection.

Examine the `/etc/ifhp.conf` file. The default configuration is for a HP family printer which supports pagecounting using PJL, synchronization using PJL, and other standard items.

If the default configuration does not meet your needs, and you do not want to use the printcap `:ifhp` entry or pass options using the `-T` command line entries, you can create your own entry in a `/... spooldir .../ifhp.conf` file. This entry should have the form:

```
option
 ...
```

For example, we might want to configure a printer that supports PCL and PostScript, needs to convert Text to PostScript, and does not have status capability. This is similar to an HP3 with a PostScript cartridge on a parallel port.

```
status@
pjl@
pcl
ps
text@
text_converter=/usr/local/bin/a2ps -o-
text_converter_output=ps
```

An alternative to this method is to create a new entry in the `/etc/ifhp.conf` file and reference this. This method has the benefit of centralizing management of the printing configuration. You usually put such an entry at the end of the `/etc/ifhp.conf` file, and it would have the form:

```
[ hp3_parallel ]
status@
pjl@
pcl
ps
text@
text_converter=/usr/local/bin/a2ps -o-
text_converter_output=ps
```

You would modify the printcap so that the model was passed either in the :ifhp information or as a command line option:

```
printer:...<newline>
  :if=/.../ifhp -Tmodel=hp3_parallel<newline>
  :of=/.../ifhp -Tmodel=hp3_parallel<newline>
  # or you can use this
  :ifhp=model=hp3_parallel
```

3.10 Extending ifhp Capabilties

One problem faced by system administrators is the need to allow users to configure various options in a job specific manner. In order to support this, the LPRng print spooler passes the `lpr -Z` command line options through to the print spooler, where they are used by the `ifhp` filter.

The details of how these options are used are explained in detail in the following sections, but here is an overview of how commands can be added.

The `pjl_user_opts=[ ... ]`, `pcl_user_opts=[ ... ]`, and `ps_user_opts=[ ... ]` are options with predefined meanings. The values of these options specify `-Z` and `-T` command line options that will be processed by the `ifhp` filter when `pjl`, `pcl`, or `ps` configuration needs to be done. Here is a sample configuration file section, and an example.

```
# we add the upper and lower keywords
#  lpr -Zupper  or lpr -Zlower will be specified by the user
pjl_user_opts=[ upper lower ]
pjl_upper=@PJL SET INTRAY=TRAY1
pjl_lower=@PJL SET INTRAY=TRAY2

#  lpr -Zpagesetup
pcl_user_opts=[ pagesetup ]
pcl_pagesetup=[ lines=12 cols=4 ]
lines=20
pcl_lines=\033*(\%{lines}L
cols=2
pcl_cols=\033*(\%{cols}C
```

If the user specifies `lpr -Zupper`, if `PJL` is available on the printer (`pjl` is set), then the `-Z` option list is scanned for an option that is also in the `pjl_user_opts`. The `upper` option is found, and then the value for `pjl_upper` is located and sent to the printer. By prefixing the context or language that is needed, we can have multiple language dependent actions for the same option.

The `lpr -Zpagesetup` option is more complex, mainly due to the range of possibilities the PCL offers. The `pagesetup` option is found in the `pcl_user_opts` list, but when `pcl_pagesetup` is found, it turns out to be a list of values.

The list values are expanded in turn, resulting in `pcl_lines` and `pcl_cols` being expanded in turn. The `pcl_lines=\033*(\%{lines}L` uses the common C/C++/Perl escape sequences, with the following addition: `\%{name}` is replaced by the value for the option name found by searching in order a a stack of values that have been set by the recursive expansion, the `-Z` options, `-T` options, and the configuration file. In this example, we would produce the output string

```
\033*(12L\033*(4C
```

For examples of how to add these commands, please examine the default `/etc/ifhp.conf`.

3.11 PCL and PostScript End of Job Sequences

One of the more troublesome problems with printing PCL and PostScript jobs is to initialize the printer when a previous job has failed. This is commonly done by prefixing a job file with a *end of job* sequence for the particular language. For PostScript, this the the Control-D (`\004`) character, and for PCL the ESCAPE E (`\033E`) sequence.

One problem with using this approach is that when you want to prefix a job with additional PCL or PostScript, you need to remove the existing end of job sequences, or the prefixed commands will have no effect. By default, the `ifhp` filter will do this. In addition, it will also prefix the new commands with an appropriate end of job sequence.

However, there are certain models of printers which do not like to have end of job sequences at the start of jobs. To suppress the prefixing of PostScript jobs with the PostScript end of job, set the `no_ps_eoj` flag, and to suppress the prefixing of PCL jobs with the PCL end of job, set the `no_pcl_eoj` flag.

4 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), massaged in the appropriate manner, and then written out 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 first nonblank input line will be treated as a request to generate a banner. The string "\031\001" will cause the filter to suspend operations using a SIGSUSP signal. At this point, job files will be sent to the output device by the spooler, and the filter will then be restarted with a SIGCONT signal.

These steps are best explained algorithmically. The following is a *pseudo-code* description of the steps performed during the printing activity. The sections marked with `###` are discussed later in this document in detail.

/// See: 4.1 (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
```

```
        }
    }
    foreach option in -Toptions, -Zoptions do
        if( option = "debug=level"  and Debuglevel not set ){
            set Debuglevel = level;
        }
        if( option = "trace" ){
            output error and trace on STDERR
        }
        if( option = "config=pathlist" and from -Toption ){
            set configuration pathlist = pathlist;
        }
        if( option = "model=name" and model not set ){
            set model = name;
        }
    }
```

/// See: 4.4 (Operation Configuration Options)

```
    // extract configuration information
    foreach path in configuration pathlist {
        open path;
        for each line in file information {
                    if( line is selected to be in configuration ){
                            process input line, adding it to configuration
                            if( line is 'debug=','model='
                               and the corresponding value not set ){
                               set the value;
                            }
                    }
        }
    }

    // get values of options with predefined meanings
    // these include status, forcestatus, etc
    foreach option in predefined list {
            if( option=value is in selected configuration information ){
                    set option=value;
            }
    }

    // open a direct connection if specified
    if( device specified using -Tdev=device ){
            // if device is host%port, we open TCP/IP connection
            fd = open(device);
            // Note - option read_write will open the device or file read write
            dup fd to 1; close fd;
    }
    if( appsocket procotol specified and TCP/IP device ){
            udp_socket = open( udp socket to device%port+1 )
    }


    ###---
```

/// See: 4.5 (Synchronization and Pagecount)

```
###+++ Synchronization and Pagecount
if( status returned by printer and sync requested ){
    // APPSOCKET protocol
    // sync has the form sync@ (none), sync=ps, sync=pjl, ...
        if( appsocket ){
                command = "\n\r"
        } else {
                // decode status=language and determine sync
        if( sync = pjl and PJL ECHO available ){
            send PJL ECHO command to printer
                } else if( sync = ps ){
            send PS program to printer
                } else {
                        terminate with error;
                }
        do{
                send command and wait for timeout;
    } while( no response );

        if( appsocket ){
                close and reopen TCP/IP connection;
        }

    // 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: 4.6 (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: 4.7 (Text File Conversion)

```
     // 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 = UNKNOWN
     if( command line -c (binary) option present ){
         language = RAW;
     } else if( -Zlanguage=xxx option present ){
         language=xxx
     } else if( file is PS file ){
         language=PS
             if( file starts with PS EOJ (CTRL-D)
```

```
                and no_ps_eoj is set ){
                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 no_pcl_eoj is set ){
                remove the PCL EOJ
            }
    } else if( file_util_path=/pathname ){
        use UNIX file utility to get file type
    }

    if( language = UNKNOWN and
        default_language option has value ){
        language = value of default_language;
    }

    if( language = TEXT ){
        if( text_converter=/path option has value ){
            run text converter on input
        } else if( printer does not support TEXT output ){
            exit with error;
        }
        language = value of text_converter_output option
    }

    if( language = UNKNOWN ){
        exit with error;
    }
    if( PJL ENTER supported ){
            use PJL ENTER command to select language;
            send nullpad NULLS to force full buffer condition
    }
```

/// See: 4.8 (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_vars=[ ... ] ){
```

```
                ###+++ expansion as above
                ###---
            }
        }
        remove whitespace and expand string results;
    } else if( language = PS ){
        ###+++ language specific actions as above,
          using the ps_ prefix for lookup
        expand string results but do not remove whitespace
    }
```

/// See: 4.9 (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 terminaton

    ###+++ Synchronization and Pagecount as above
    if( waitend ){
            if( sync requested previously ){
                    if( sync with PJL ){
                            wait for end of job using UINFO;
                    } else if( sync with PS ){
                            request status using ^T and wait for
                            printing to stop
                    }
            }
            if( appsocket ){
                    close and reopen connection;
            }
            get pagecount using previously descibed algorithm
    }

    ###---

    exit
```

4.1 Options, Initialization and Setup

The `ifhp` filter is designed to work with the LPRng print spooler, but will also work with other spooling systems. The LPRng system will set the `PRINTCAP_ENTRY` environment variable to the current printcap value. By convention, the filter command line `-Toptions` are reserved for the print spooler to pass configurtion inforamtion and the `-Zoptions` are passed by the user. For example, examine the following `lpr` command and printcap example:

```
    Printcap:
```

```
pr:...
  :ifhp=opt1=value1,opt2=value2
  :if=/usr/ifhp -Topt1=value4,opt3=value3

Command:
lpr -Zopt4=value4

PRINTCAP_ENTRY envionment variable:
  pr:...\n
    :ifhp=opt1=value1,opt2=value2\n
    :if=/usr/ifhp -Topt1=value4,opt3=value3\n

Resulting option list:

-Toptions:
  opt1=value4
  opt2=value2
  opt3=value3

-Zoptions:
  opt4=value4
```

When started, the `ifhp` filter process the environment and command line options as follows.

1. If the `PRINTCAP_ENTRY` environment variable has a value, then this value is used to initialize the `-Toption` list.
2. If there is a `-Toption` command line option, then these values are added to the option list, overriding values from the `PRINTCAP_ENTRY` set.
3. The command line `-Zoption` list is generated by splitting the `-Zoption` command line option.

The option lists are scanned for values for the `debug`, `trace`, `config`, and `model` options. These options have the property that once they are set, then they cannot be modified (i.e. - sticky values).

There is another, and rather bizarre way to specify the printer model, and that is the `model_from_option` option in the configuration file. This option causes the command line options to be scanned, and if there is a value for the command line option then it is used as the model. For example:

```
model_from_option=Q
```

The above setting will cause the model to be taken from the `Q` option. This can be used to select a configuration for the printer based on values specifed by the user.

4.1.1 Debug and Trace

The value of the `debug` option sets the debugging level. It can be increased, but not decreased. The `trace` flag causes debugging information to be sent to STDERR (file descriptor 2) as well as to the status file.

4.1.2 Configuration File Paths

The main source of configuration information are the configuration files. The

```
config=pathname,pathname
```

option can be used to specify the list of configuration files to be read. This can only be done using the PRINTCAP_ENTRY :ifhp entry or the -Tconfig=pathname,pathname command line option.

4.2 Model Selection

The model=name option is used to establishes the model name for extracting configuration information. For details on this, see 7 (Configuration Files).

During initialization, the -Toptions list is scanned for a -Tmodel=name entry. Once the model name is set, it cannot be changed. After this, the configuration files are read, and the first model=name option encountered will set the model option to name.

The recommended method of model selection is to specify it in the LPRng printcap entry for the printer, using the :ifhp=... printcap field. For example:

```
lp:ifhp=model=HP4,status@
  :if=/usr/local/ifhp

Resulting -Toption List:
 model=HP4
 status@
```

This will cause the -Toption list to be initialized as indicated, and the model option value will be set to HP4.

The next method is use a -Toption command line option.

```
lp:...
  :if=/usr/local/ifhp -Tmodel=HP4

Resulting -Toption List:
 model=HP4
```

This will cause the -Toption list to be initialized as indicated, and the model option value will be set to HP4.

Another method is to put the model information in a ./ifhp.conf file in the spool directory of the print queue. The config=/pathname,/pathname,... option specifies the list of configuration files to read, and the default value is:

```
config=./ifhp.conf,/etc/ifhp.conf,./ifhp.conf
```

If the model information is put in the ./ifhp.conf configuration file, the first reading will set the model name, and the name is used to select the model information from the /etc/ifhp.conf file. When the ./ifhp.conf is reread, the values in it can be used to override values from the /etc/ifhp.conf file. For example:

```
./ifhp.conf:
  model=HP4
  lines=66

/etc/ifhp.conf:
[ hp* ]
 lines=60
[ apple* ]
 lines=20
```

When the ./ifhp.conf is first read, it will establish model=hp (sticky) and lines=66. When the /etc/ifhp.conf file is read, the model name matches the hp* selector (case insensive GLOB matching is used), and the lines=60 is selected and overrides the lines=66 value. Finally, when the ./ifhp.conf file is reread, lines=66 will establish the final value.

4.3 Statusfile, Statusfile_max, Statusfile_min

The status file pathname is set by the command line -s /pathname or the statusfile=/pathname configurtion file entry. If the /pathname file does not exist then it will not be created. If the statusfile is larger than the statusfile_max=max K bytes option value (default 8K), then it will be truncated to statusfile_min=min K bytes.

4.3.1 Summmaryfile

For historical and vintage software compatibility, the summaryfile=/pathname or summaryfile=host%port option will cause either a file to be open or a UDP network connection established to the host and port combination. Debugging or trace information will be written to this file or network connection as well, but the file will be truncated each time, holding only the last line of trace information.

4.4 Operation Configuration Options

The -Toption=value and model configuration information is scanned to set values of options which control filter activity. There are some options whose related actions do not fall into the simple model of string expansion. These usually require generating commands dynamically, or sending files containing font or setup information to the printer. The following is a list of these options.

4.4.1 status and forcestatus FLAGS

These options have the side effect of enabling the reception of status and error information from the printer.

4.4.2 pjl, pcl, ps and text FLAGS

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

4.4.3 crlf FLAG

The crlf causes LF (\n) to be translated to CR-LF (\r\n) sequences. The following options will turn the ifhp filter into a simple CRLF translation filter. Note that CRLF translation should have no effect on PostScript, Text, or PCL files.

```
status@
pjl@
ps@
pcl@
text
text_converter_output@
text_converter@
crlf
```

4.4.4 pjl_job FLAG

If PJL is enabled and this flag is SET, a PJL JOB and PJL EOJ command will be generated and sent to the printer. The JOB command has the form:

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

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

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

4.4.5 pjl_enter FLAG

If PJL is enabled and this 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
```

4.4.6 nullpad STRING

Some older model HP printers required sending 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 historical, as most printers do not have this problem.

4.4.7 pjl_console FLAG

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

4.4.8 remove_ctrl STRING

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.

4.4.9 tbcp FLAG

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
```

4.5 Synchronization and Pagecounts

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` or `forcestatus` option is TRUE, then the filter assumes that it can read FD1. In order to simplify configuration, the `ifhp` filter will test FD1, and if it is not a serial port or a network socket, will set `status@` or OFF.

However, there are some devices such as bidirectional printer ports that will report status. By setting `forcestatus` ON, the filter can be forced to check for status. This can have fatal or unexpected effects if status is not returned correctly.

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 usally 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 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.

Pagecounts are used to do accounting and report the number of pages used for a job. Most printer 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
```

Pagecounting is supported by the `pagecount=`*language*, `pagecount_interval=nnn`, and `pagecount_timeout=nnn` options. The `pagecount=`*language* option enables pagecounting, and sets the language to be used. Currently `ps` (PostScript) and `pjl` (PJL) are supported. The pagecount request is repeated 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. A 0 value or `sync_timeout@` disables timeouts.

Some printers do not correctly report end of job and must be polled until the pagecount information stabilizes. The PJL TEOJ (True End Of Job) PJL has been tried with limited success on various printers to force End of Job reporting only when the job has finished.

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

4.6 PJL Initialization

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 Lanaguage (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 alorithm 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
   ```

4.7 Text File Conversion

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. Also, while most PCL printers will accept text files and do a reasonable job of printing them some form of initialization strings or setup may need to be done. Finally, you might want to try using a `MagicFilter` that will convert just about any type of file into a PostScript file.

4.7.1 Default Converter Program

This method specifies that the default file type will be `text`. If the simple `ifhp` type detection code cannot decide what type of file this is, it will invoke a converter program. This operation is controlled by the following options.

`default_language=text`

When the default lanague is `text` then the `text_converter` and `text_converter_output` options are used.

`text_converter=/pathname`

specifies the pathname of a text to language program.

`text_converter_output=language`

Specifies the output language of the conversion program. Language can be `ps`, `pcl`, `text`, `raw`, `unknown`

`tempfile=/pathname`

A temporary file location, used to store intermediate conversion results. The `/pathname` value has the string `XXXXXX` appended to it and is used as input to the POSIX `mktemp()` function.

The print file language is determined using the following algorithm.

1. The default language is set to `unknown` or the `default_language=language` value if it exists.
2. If the command line `-c` (binary) option is present, or the `autodetect` configuration option is TRUE, then the language is set to `RAW`. (The autodetect option is not recommended for general use).
3. If there is a `-Zlanguage=value` command line option, the language is set to `value`.
4. Various simple checks to determine if the file is Postscript (language=`ps`) or PCL (language=`pcl`) are performed. These are the same checks that the UNIX `file` utility uses.
5. If the file type is `text` then the `text_converter` program is run to translate the file and the results stored in the `tempfile`.

One technique used with varying degrees of sophistication is to use a general purpose file to PostScript conversion program. These have generally been known as *MagicFilters*, due to their high degree of flexibility.

4.7.2 Pseudo-MagicFilter Support

As described above, the various *MagicFilter* packages can do conversion. However, most of the time there is only a limited need for the general purpose conversions. This can be met by using the `file` program, which will determine the type of a file based on its contents, and having `ifhp` invoke a program based on the type of file found. This method is used when `default_language=unknown` and the `file_util_path=/path` and `file_output_match=[ ... ]` values are defined. A typical configuration is shown below.

```
Configuration:
#
#  Method 2 -
#    Use the file util and match the output
# file reports format information
#  glob text_output_format text_converter
#  - you do a glob match against pattern and use the converter
#
default_language=unknown
file_util_path=/usr/bin/file -
file_output_match = [
# glob   output converter
 *text*  ps     /usr/local/bin/a2ps -q -B -o-
 *gif*   ps     /usr/local/bin/gif2ps
 ]
```

The `file_output_match` entry consists of a list of lines containing a *glob* pattern, a language type (`ps`, `pcl`, or `raw`), and a program to invoke to do the conversions.

The `file` utility is run and its output is matched against the specified glob patterns. When a match is found the specified program is run, with STDIN attached to the original input file and its STDOUT sent to the printer.

While this algorithm may appear to be overly complex, it will handle a wide range of desired configurations.

4.7.3 No Textfile Conversion Needed

If your printer can handle text files without conversion, but require PCL intialization, then the following combination will simply set the language to `pcl`:

```
text_converter@
text_converter_output=pcl
```

4.7.4 Default Passthrough of Unknown File Types

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:

```
text_converter@
text_converter_output=raw
```

4.8 Language Specific Initialization

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.

4.9 File Transfer and Error Status Monitoring

If the printer can return status, i.e., the `status` or `forcestatus` flag is set, then the `ifhp` filter will read status information back 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
DISLAY="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. For example:

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

Also, there may be error codes which does 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 corresonding 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
  ]
```

4.10 End of Job

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 using the `appsocket` protocol, then suppressing `waitend` will cause no error messages from the printer to be reported.

4.11 Tektronix Phaser and AppSocket Support

The Tektronix Phaser PostScript printers uses the AppSocket protcol for sending a job to the printer over a network connection. The `appsocket` flag enables this operation. The protocol is (briefly):

1. The printer listens for UDP packets on port 9101 and for TCP/IP connections on port 9100.
2. When a UDP packet is recieved on port 9101, then a reply packet containing the status is returned to the originator's address. This packet contains an status indication, in a *undefined* format.
3. To send a job to the printer, a TCP/IP connection is opened to port 9100, and a PostScript job is sent. Only a single job can be sent at a time - a EOJ (CTRL-D) will terminate input and flush all following jobs.
4. Return status will be sent in the reverse direction until the job has completed, at which point the connection will be closed.

To use this protocol, the printer TCP/IP address and port must be specified using the `-Tdev=host%port` option; usually port is 9100. Also, the printer device in the printcap entry should be `lp=/dev/null`.

When using the Appsocket protocol, the `ifhp` filter will open a UDP port and use it to send query packets to the printer UDP port 9101. In addition, it will try to open a connection to port 9100. When a connection has been established, and pagecount has been determined, the connection will be close and reopened.

After job transfer, the connecion will be `half-closed`. That is, the `shutdown()` facility will be used to cause the TCP/IP connection to be set to closed for transmission but open for reception. The printer will send status information until the job is completed, and then close the connection.

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

5 Banners and OF Mode Operations

The OF mode is enabled by the command line -Fo flag or by the file name of the executable containing the "of" string.

When operating in the OF mode, the filter uses the value of the banner key to determine what to do with input. If banners are disabled (`banner@`), then input is simply passed directly to the output. The banner_suppressed flag allows you to suppress banner printing until explicitly requested by a user using the -Zbanner flag.

`banner=pcl`

Uses a built-in PCL banner generator. Pretty simple output.

`banner=ps`

Using information from the command line options, generates PostScript line which set the values of PostScript variables. Then the contents of the file specified by `banner_file=/path` are appended.

```
%!PS-Adobe-2.0
/Seq (number) def
/Job (banner) def
/Host (HOST) def
/Class (CLASS) def
/User (USER) def
/Date (DATE) def
/Name (NAME) def
/Line (LINE) def
/X (command line -X option value) def
```

`banner=text`

Puts out a simple text based banner. Uses the command line `-w width` and `-l length` command line options to set the page width in columns and pagelength in lines. If these are not set, the configuation options `page_width=width` and `page_length=length` values are used.

`banner=/path`

Opens and copies the file directly to the printer.

`banner=|/path`

Forks and EXECs the executable specified by `/path`, with the same arguments as those passed to the filter. The executable output is sent to the printer.

6 Predefined Options

The following is a list of predefined options.

4.11 (appsocket FLAG)	Use Tektronix AppSocket Protocol
5 (banner=LANGUAGE)	Enable OF banner printing using LANGUAGE
5 (banner_file=PATHNAME)	PATHNAME of PostScript banner file
4.1.2 (config=PATHNAMES)	Configuration file pathnames
4.4.3 (crlf FLAG)	Do LF to CRLF translation
4.1.1 (debug FLAG)	Debugging level
4.7 (default_language=LANGUAGE)	Default job file language (ps, pcl, raw, text, etc)
4.6 (endpage=NNN)	PJL JOB command END = NNN value
4.7 (file_util_path=PATHNAME)	Pathname of the UNIX file utility
4.4 (forcestatus FLAG)	Force status reading from the printer if set
4.7 (language=LANGUAGE)	Specify job file language to be used (ps, pcl, raw, text, etc)
4.9 (logall FLAG)	Log all status reports from printer if set
4.4.6 (nullpad=COUNT)	Send COUNT nulls to force full buffer condition
4.2 (model=NAME)	Specify model name for configuration selection
4.1 (model_from_optin=X)	Specify model name using a command line option value
3.11 (no_ps_eoj FLAG)	No PostScript EOJ (CTRL-D) at start of job
3.11 (no_pcl_eoj FLAG)	No PCL EOJ (CTRL-D) at start of job
5 (page_length=LINES)	Number of lines on a text page for banner printing
5 (page_width=COLUMNS)	Number of columns on a text page for banner printing
4.5 (pagecount=LANGUAGE)	Enable pagecounting using the specified lanague (pjl, ps)

4.5 (pagecount_interval=SECONDS)	Send pagecount command at SECONDS interval
4.5 (pagecount_ps_code=STRING)	PostScript code to get pagecount information
4.5 (pagecount_timeout=SECONDS)	Timeout getting pagecount after SECONDS
4.4 (pcl FLAG)	Printer supports PCL if set
3.10 (pcl_user_opts=LIST)	User PCL options supported
4.4 (pjl FLAG)	Printer supports PJL if set
4.4.7 (pjl_console FLAG)	Printer supports messages on console
4.9 (pjl_error_codes=LIST)	PJL error messages for error codes
4.6 (pjl_except=LIST)	Do not allow these PJL commands
4.6 (pjl_init=LIST)	PJL initializations to be done
4.4.4 (pjl_job FLAG)	PJL JOB and EOJ supported
4.6 (pjl_only=LIST)	Allow only these PJL commands
4.6 (pjl_user_opts=LIST)	Allow only these user PJL commands or variables to be set
4.6 (pjl_vars_except=LIST)	Do not allow these PJL variables to be set
4.6 (pjl_vars_set=LIST)	Allow these PJL variables to be set
4.4 (ps FLAG)	Printer supports PostScript (ps)
3.10 (ps_user_opts=LIST)	Support these PostScript user options
4 (read_write FLAG)	Open a file or device read/write
4.4.8 (remove_ctrl=LIST)	Remove these characters from PostScript jobs
4.6 (startpage=NNN)	PJL JOB command START = NNN value
4.4 (status FLAG)	Printer supplies status information
4.3 (statusfile=PATHNAME)	Status file pathname
4.3 (statusfile_max=NNN)	Status file has maximum size of NNN Kbytes
4.3 (statusfile_min=NNN)	Status file has truncated size of NNN Kbytes
4.3.1 (summaryfile=PATHNAME)	Summary file pathname
4.5 (sync FLAG)	Synchronize printer if set
4.5 (sync_interval=SECONDS)	Send synchronization request at SECONDS interval
4.5 (sync_timeout=SECONDS)	Timeout synchronization request after SECONDS
4.4.9 (tbcp FLAG)	Use Transparent Binary Communications Protocol for PostScript files
4.4 (text FLAG)	Printer supports text mode
4.7 (text_converter=PATHNAME)	Pathname of text conversion program
4.7 (text_converter_output=LANGUAGE)	Job language of text converter output
4.7 (text_tempfile=PATHNAME)	Temporary file pathname template
4.1.1 (trace FLAG)	Put error and trace messages on STDERR if set

7 Configuration Files

Run time options are provided by the command line arguments and values in the configuration files. By convention, the configuration files are named `ifhp.conf`.

In order to provide a flexible run time configuration facility, the location of the configuration files is specifed as follows.

1. The default list of configuration files is:
   `./ifhp.conf,/etc/ifhp.conf,./ifhp.conf`
2. The command line option
   `-Tconfig=/path,/path,...`
   can be used to override the default list of configuration files.

Here is a section of a simple configuration file.

```
# languages
pcl
statusfile=status
status
sync=pcl
sync_interval=20
# we force pagecounting off
#pagecount=pcl
pagecount@
[ HP4Si ]
status@
end
[ HP5Si ]
pjl
sync=pjl
```

The configuration file is used to set flags, option values, and to cause various `ifhp` actions. The file has the following structure.

1. Blank lines and lines starting with # are ignored.
2. Keys or flags start in column 1. The syntax is similar to the LPRng and BSD file.

   ```
   Syntax            Equivalent To    Class
   flag              flag=1           (FLAG)
   flag@       flag=0           (FLAG)
   flag=val          flag=val (string value) STRING
   flag=] v v  ... [          LIST
   ```

3. Flags are used for options which take a TRUE/FALSE, 0/1, or ON/OFF value, and `ifhp` will substitute the appropriate form or perform the assocated action if the flag is TRUE.
4. Strings are used to set options which require a multicharacter value. The special form `flag@` is used to indicate that the operation related to this option is not to be performed.
5. 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 items in the list will be acted upon in order. This is discussed later in detail.
6. The list entries are separated by whitespace, and each entry can have the form v, v@, v=word, or v1#word, where word does not contain whitespace or the [] characters.
7. Flag values can be spread over multiple lines. Lines starting with whitespace, are treated as a continuation of the previous flags line value. For example:

   ```
   # set string x value 'first\n  second\n  third'
   x= first
     second
     third
   # set list y value [ f1 f2 ]
   y=[ f1
    f2 ]
   ```

8. Selection lines have the form:

   ```
   [ glob glob ... ]
   ```

Selection lines divide the configuration file into sections corresponding to a particular printer model. Configuration information is extracted from a file until either an 'end' line or a selection line is encountered.

9. An 'end' line consisting of the single work 'end' will terminate reading lines from a configuration file.
10. As the configuration file is read, flag lines and values are accumulated. Later values encountered in the file will replace earlier values.

7.1 Configuration Selection

The recommended format for a configuration file is to put common (default) flag settings at the start of the configuration file, followed by selection sections with overridding and additional flag values.

To allow a single file to be used for multiple printer configurations, you can specify that a section of the file is to be used ONLY by a various models of printers. This is is controlled by the value of the 'model' flag and selection lines of the form:

```
[ glob glob ... ]
```

The first occurrence of a 'model=xx' line in either the -T options or the configuration file will set the model flag value to 'xx'.

The 'model' value is matched against the modelglob values using GLOB matching. For example:

```
hp*      matches hp4 hp5x
hp[45]   matches hp4 hp5, but not hp5x
hp[3-6]* matches hp3, hp5, hp5x, but not hpiii
```

If a matching entry is found, successive lines will be used until either another selection line or an 'end' line is encountered. An 'end' line will terminate reading the current file, and the next configuration file will be read.

The default list of configuration files is:

```
ifhp.conf, /etc/ifhp.conf, ifhp.conf
```

This arrangement allows you to read the ifhp.conf file to get various model settings, scan the /etc/ifhp.conf file to get the generic ones, and then to rescan the local ifhp.conf file to provide overrides to values set in the /etc/ifhp.conf file.

7.2 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 are simply used to supply values for expansion during operation.

The builtin options are listed in later sections, and their use is explained. These options can have flag, string, or list values as is appropriate to their corresponding actions.

7.3 Recursive List Expansion

During normal operation, the `ifhp` filter will perform an operation by producing a set of strings which will be sent to the printer or output device. These strings may be obtained by using the values of predefined or builtin option names, or by expanding a LIST value.

A LIST value has the form X=[v1 v2 ...]. When a list value is expanded each of v1, v2 is examined in turn and the corresponding action or string substitution or builtin evaluation is carried out. If v1 has a string value and is not recognized as a builtin or special option then normally the string value will be used.

```
t1=[ p1 p2 p3=end ]
p1=this is
p2=[ p3 p4 ]
p3=a
p4=test
p3=living\020\%{p3}
```

For example, when expanding `t1` each of `p1` and `p1` is in turn expanded. This will produce the strings `"this is"`, `"a"`, `"test"` and `"living end"` in turn.

Some LIST variables are used in printer language specific contexts and their values are processed appropriately. For example, pjl_init=[...] specifies a set of operations to be carried out for printers that support PJL, and pcl_init=[...] for PCL printers. 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. For this reason, during expansion the language name and an underscore is prefixed to the list entry name, and this is used as the option name during the search.

For example, suppose that we have:

```
pjl_init=[ test ]
pcl_init=[ test ]
initstr=NO
pjl_initstr=@PJL ECHO YES
pcl_initstr=\033(*yeS
```

When PJL initialization is done, the 'pjl_test' LIST will be expanded, and the PJL string '@PJL ECHO YES' will obtained. When PCL language specific processing is done, then the \033(*yeS string will be obtained.

7.4 String Expansion

String values are encoded using a simple PERL/C language like method. The \ (escape) character introduces a replacement string. This has the form:

Standard Character Replacement

`\f \r \n \t \nnn` where nnn are 3 octal digits are replaced by the standard PERL or C character substutions.

Option Value Replacement

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

The option name will be relaced by the formatted option value.

Option Search Order

The option value is located using a simple set of rules.

1. During a recursive option evaluation, expanding `option=word` will push the `option=word` combination onto an evaluation stack. This stack is searched in oldest to newest order for a match.
2. If no match was found, and the expression has the form {`option`} then the `-Zoption=value` command line options will be searched for a match.
3. If no match was found, then the `-Toption=value` command line options will be searched for a match. This allows the {`option`} to start searching from the -Z command line options and `[option]` to start searching from the -T command line options.
4. If no match was found, then the configuration information is searched.
5. If no match was found, then the value is considered undefined, and a `"0"` value is used.

Format

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

```
%[-][0][length[.precision]][format]
```

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 option string value.

The format is usually not required, except when fractional values of point sizes or string substition rather than numerical substition is required.

For example:

```
Configuration:
  pjl_user_opts=[ ... outbin intray ...]
  pjl_outbin=@PJL SET OUTBIN=\%s{outbin}
  intraynum=4
  pjl_intray=@PJL SET INTRAY=\%{intraynum}

Command:
  ifhp -Zoutbin=LEFT
```

During PJL language processing, the `-Z` command line options will be scanned for options which appear in the `pjl_user_opts` list. The `-Toutbin=LEFT` option will be found and will be expanded in the PJL context by prefixing `pjl_` and looking for a string or list value. The `pjl_outbin` option will be found, and the `@PJL SET OUTBIN=\%s{outbin}` string will be expanded.

Now we need to search for the `outbin` value. We first search for it on the evaluation stack, but there is nothing there yet. We then search the `-Z` options and find the `outbin` value, and substition yields `@PJL SET OUTBIN=LEFT`.

Next, the `intray` option is found and `pjl_intray` is expanded, which needs a value for `intraynum`. This is found in the configuration information, and finally in `@PJL SET INTRAY=4`.

```
cpi=5.5
pcl_cpi=\033\%3.2f{cpi}D
```

During PCL option expansion, we might need to expand the `pcl_cpi` option. When the `pcl_cpi=\033\%3.2f{cpi}D` string is expanded, the result is `\033\%5.00D`. If the user has specified `-Tcpi=9` on the command line then the result is `\033\%9.00D`.

7.5 Language Context

The Tifhp filter sends initialization and configuration commands to the printer. Depending on the language, these commands have specific forms and requirements. Rather than requiring the user to remember the details, Tifhp uses the following conventions.

7.5.1 PJL Language

A PJL command has the form `@PJL OPCODE ...`. A command must start with `@PJL` and consist of a single string value. You cannot patch together options to make a single PJL command.

1. Before sending any PJL command to the printer, the PJL Universal Exit Command (`\033%-12345X`) string is sent to the printer.
2. Because not all printers support all PJL commands, the Tifhp filter performs a couple of checks using the `pjl_only` and `pjl_except` configuration options. The OPCODE must appear in the `pjl_only` list and not in the `pjl_except` list.
3. Leading and trailing whitespace is removed, and all characters are converted to uppercase.
4. Individual commands have a newline (`\n`) appended to them before being sent to the printer.

7.5.2 PCL Lanaguage

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 processing PCL strings.

1. Before any PCL string is sent to the printer, the PCL End of Job (`\033E`) string is sent to the printer.
2. First, 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.

7.5.3 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.

8 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.

8.1 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 intitialization 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.

8.2 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.

8.3 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.

9 Banner Printing

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.

9.1 Banner Printing with LPRng

The following steps need to be done to configure LPRng to print banners using `ifhp`.

1. In order to print a banner, the LPRng print spooler will require a user name for the banner page to be present in the print job. However, since users can request *no banner* via various options on the print spooler interface, LPRng provides the handy *force_ banner* option to always force a banner to be generated, even if the user has specifically requested that one *not* be generated. This is useful for preventing the *disappearing print job* syndrome in large installations.

2. Next, LPRng must be configured to generate banners. The `sh` (suppress header) option must be off (`sh@`).

3. At this point you need to decide if you want LPRng to generate the banner of the `ifhp` filter to generate the banner. If you want LPRng to generate the banner, you can specify a `banner` generation program, and inform `ifhp` **not** to generate a banner (the default).

```
Printcap:
lp:
  :sh@
  :banner=/usr/local/filters/bannergen
  :if=ifhp -Tbanner
```

4. If you want `ifhp` to generate a banner, you do not have LPRng generate a banner and enable banner printing by `ifhp`. You also can specify the `sb` or short banner option, which will cause a very short dummy banner to be generated. This will be ignored by the `ifhp` filter.

```
Printcap:
lp:
  # short banner, save effort
  :sh@:sb
  :if=ifhp -Tbanner
```

5. Finally, you have your choice of PCL, PostScript and even Text banners. These can be specified using `banner=`**language**:

```
Printcap:  lp:
# short banner, save effort
:sh@:sb
:if=ifhp -Tbanner=language
```

9.2 Stand Alone Banner Program

Occasionally it is useful to be able to generate a banner in *standalone* mode. For example, you might want to generate a banner when using an LPRng *bounce queue* to perform filtering operations before forwarding jobs to another printer. To do this requires a stand-alone banner printer. `ifhp` can be configured to do this by using:

```
ifhp -Tbanner_only
```

In addition, you can specify the type of banner you want using:

```
ifhp -Tbanner_only=language
```

Lastly, in order to be *vintage software compatible*, if the ifhp program is invoked as the `banner` program, this is effectivly the same as

```
ifhp -Tbanner_only
```

For example, to use this with a bounce queue to an `hp4` printer, the following printcap entry might be used:

```
# for clients, force spooling to server
lp:lp=lp@server
```

```
# server
lp:server
  :bq=raw@printer
  :ifhp=model=hp4
  :generate_banner
  :banner=/usr/local/filters/ifhp -Tbanner_only
  :if=/usrlocal/filters/ifhp
  :of=/usrlocal/filters/ifhp
```

10 Accounting

In Academic institutions, avoiding printing accounting has been regarded as a challenge, an ongoing game of fat cat and poor starving mouse, between the Adminstration and the downtrodden, poor, overcharged student. The following is a lighthearted ramble down the dark lane of printing accounting.

We will disregard the fact that if most students put as much effort into their studies as in finding ways to avoid accounting procedures then they would be Rhodes Scholar material, but I digress...

10.1 Page Accounting Algorithm

The accounting procedures put into the LPRng and the ifhp filters may appear to be extraordinarily complex, but believe me, they are not. Firstly, we make the assumption that the printer has some sort of non-volatile page counter mechanism that is reliable and impervious to power on/off cycles. Without this mechanism the enterprising student ummm... user will simply turn off the printer. Software that prescans jobs for line counts and pages is notoriously unreliable, given even the most modest efforts of users to hide these procedures. The cost of running a PostScript simulator simply to do accounting has its flaws; without ensuring that the simulator has all of the interesting security loopholes closed, such as opening files, etc., it can become a trap door to hell for the system administrator.

Secondly, we must make the assumption that the student... uhhh... user will not be able to tinker with the page counter mechanism, i.e.- they will not be able to roll back the odometer on the printer, *for the duration of a single job*. I will digress and point out that a student actually did this for a challenge; it only took him a couple of weeks of study and a fully equipped microcontroller lab, and two (2) laser printers which he ruined in the experiment. HP was not amused when we sent them back under warranty, claiming that this our 'normal lab usage.'

Lastly, you should not mind a small amount of pilferage, or a few pages here and there being charged to the wrong account.

The basic mechanism the ifhp filter uses is to record the page counter value at the start and end of each part of a print job. Each record has the form:

```
start -qpagecounter ....
end  -ppages -qpagecounter -telapasedtime ....
```

When we use the OF filter and/or banners, we will see the individual jobs bracketed by the OF filter records:

```
start -q100 -Fo -kcfA100taco -uuser -hhost -R...
start -q101 -Ff -kcfA100taco -uuser -hhost -R...
end  -p1 -q102 -Ff -kcfA100taco -uuser -hhost -R...
start -q102 -Ff -kcfA100taco -uuser -hhost -R...
end  -p3 -q105 -Ff -kcfA100taco -uuser -hhost -R...
end  -p5 -q105 -Fo -kcfA100taco -uuser -hhost -R...
```

It should be clear from the above that all we need to do is to add up the values for the -Fo (OF) filter lines and we are done.

Unfortunately, this is too simplistic. If for some reason the job is killed or terminates due to error conditions, the OF filter may not get to finish its work. Thus, we may see the following:

```
start -q100 -Fo -kcfA100taco -uuser -hhost -R...
start -q101 -Ff -kcfA100taco -uuser -hhost -R...
start -q110 -Fo -kcfA101taco -uuser -hhost -R...
```

This is a clear indication that the user's job has been terminated. In this case we need to use the differences between pagecounters of the start records to do accounting.

There is a caveat to all of this; that is the problem of the last dead job in the list. If the last line in the accounting file is:

```
start -q110 -Fo -kcfA101taco -uuser -hhost -R...
```

is the last job finished or did it abort?

10.2 You Used 2000 Pages, Out of Quota

Now we move on to the problem of real time accounting. Due to limited budgets, etc., many institutions would like to strictly enforce limits on paper use by students. As jobs are printed their accounts should be docked for the amount of paper use. One way to do this is to have an external accounting procedure update a shared database. The ifhp filter has provision for a shell script to be invoked at the start and end of print job; this is done by both the OF and IF filter. Thus, we can blithely assume that there is a central database carefully getting updates from the LPRng software, probaly from dozens of different printers, and updating the accounting information, and that this program can query the database, check limits, and terminate printing if the limits are exceeded.

The first question to be asked is simple: is this worth it? Perhaps doing accounting as a batch job once an hour/four times a day/once a day is cheaper than buiding an running such a database. If it costs $5K/year for the database software, you might just consider ignorng the 10,000 pages that get lost in the shuffle and use a simple set of awk/sed/perl scripts to update a database once an hour.

10.3 Bad Jobs - Who Do We Bill?

We inevitably run into an interesting question: what happens if a job does not complete correctly?

If you use the completion of the OF filter as a success status, I have to point out that many students... ummm... users soon find ways to send jobs to the printer that will cause it to lock up after their output has been printed. These jobs require power cycling of the printer and restarting the filter; a bit extreme, perhaps, but it has happened.

I suggest that you simply adopt a 'bill to last user of record' attitude, using the pagecount information as follows:

```
start OF -- starting point for THIS job
start IF --  nice information, but not useful
start IF --
end OF -- ending point for this job - can record infomaiton

start OF --
```

```
if no end OF for previous job,  then treat as end OF and
          update accounting.
```

Now somebody is sure to complain that they got charged for a bunch of pages that they did not use. This is inevitable; always carry a can of oil for the squeaky wheels. I might make the observation that once is accident, twice is coincidence, but three times is malice; be wary of the constant complainer and check out not only him but also his co-workers.

10.4 How Do We Update The Database

I suggest that database update be done as follows: maintain a 'last page reported' record for each printer in the database.

When a successful job reports in, check to see that the recorded pagecount for the printer is in agreement with the one that is reported.

If this is not the case then you have had some unsuccessful jobs. In this case I strongly recommend that you have a means to request the accounting reporting program to go back through the accounting file and find the last report for the page counter value and try to backtrack through the accounting files. The accounting file is one of the first things to be attacked by students... Ummm... users. It should NOT be kept on and NFS exported or mounted file system. It should be carefully pruned and copied, perhaps on an hourly basis.

Now some adminstrators have fallen in love with network based printers; do not believe ANYTHING that comes over a network connection without some form of authentication; PGP has some very nice Public Key mechansims for handling this. This is a major weakness in using a database for keeping track of accounting - a weak authentication mechanism may lead to denial of service attacks by students flooding the database with bogus print usage reports; suddenly NOBODY can print and the adminstrator is driven to turning off accounting.

Good luck. I am never suprised when I encounter yet another wrinkle in this area.

10.5 Accounting Shell Script

The

```
accounting=/pathname
```

specifies a program to run at the start and end of the `ifhp` activity. For an example of a simple script, see the accounting.sh script in the distribution.

10.6 Pagecounter Values

The only reliable way to do page counting in the face of PostScript, PCL, and other mystical printer job languages is to query the printer and get the current value of a hardware page counter. Unfortuately, this may not be a trivial matter.

1. The page counter may not be updated in real time. This means that you may need to wait a couple of seconds until you are sure that the pages have been recorded. Usually this occurs when the pages leave the print engine and are put in the output tray.

2. Printers do job batching, and when they report job completion the job is still being printed.

3. Some printers report *impressions*, i.e.- sides of pages printed, rather than pages. If you are doing duplex printing then you may find that your paper count and your page (impression) count differ.

4. Some printers simply lack page reporting.

Given these problems, it is more than reasonable to reconsider the need for accounting, or to work closely with the printer vendor to understand the interactions of print jobs and reporting of page counts.

11 Monitoring Printer Operation

Normally the `ifhp` filter will write error and trace messages to the `statusfile` specified by the command line `-s` option or the `statusfile=pathname` configuration entry.

However, there may be a need to get information sent to a central or other location for system administration purposes. The `summaryfile=host%port` option will cause `ifhp` to open a UDP connection to the specified port on the remote host and send the error and trace information to that location as well.

The `monitor` program include in the LPRng and `ifhp` distributions is a template for writing a program to use this information.

12 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.

12.1 TCP/IP Network Address

You can set the network address from the front panel. Reset the printer; 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.
```

12.2 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
```

12.3 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.