For Apple Personal LaserWriter NTR printer

Personal LaserWriter NTR Printer

Developer Note

Developer Note
030-2664-A

Developer Technical Publications
Apple Computer, Inc. 1992

APPLE COMPUTER, INC.

The Apple logo is a registered trademark of Apple Computer, Inc. Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014-6299
408-996-1010

Apple, the Apple logo, APDA, AppleTalk, LaserWriter, LocalTalk, and Macintosh are trademarks of Apple Computer, Inc., registered in the United States and other countries.

TrueType is a trademark of Apple Computer, Inc.

Adobe, Illustrator, and PostScript are trademarks of Adobe Systems Incorporated, which may be registered in certain jurisdictions

AMD is a registered trademark of Advanced Micro Devices Corporation.

IBM is a registered trademark of International Business Machines Corporation.

ITC Garamond and ITC Zapf Dingbats are registered trademarks of International Typeface Corporation.

LaserJet IIP is a registered trademark of the Hewlett-Packard Company.

Microsoft and MS-DOS are registered trademarks of Microsoft Corporation.

Varityper is a registered trademark of Varityper, Inc.

Simultaneously published in the United States and Canada.

LIMITED WARRANTY ON MEDIA AND REPLACEMENT

If you discover physical defects in the manual or in the media on which a software product is distributed, APDA will replace the media or manual at no charge to you provided you return the item to be replaced with proof of purchase to APDA.

ALL IMPLIED WARRANTIES ON THIS MANUAL, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.

Even though Apple has reviewed this manual, **APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Figures and Tables

Preface

The Personal LaserWriter NTR printer is the newest member of the LaserWriter printer family. This developer note provides information that describes the features and capabilities of the Personal LaserWriter NTR printer.

To use this note, you need to understand the Adobe™ PostScript™
Level 2 programming language, and you should be familiar with the computer for which you intend to develop software.

You do *not* need to use this note when simply running packaged
programs for your Apple computer. However, this note can help you
if you are writing or modifying a program that uses the Personal LaserWriter NTR printer.

Instructions for connecting the printer to your computer, inserting paper, and performing other routine operating tasks are not provided in this note. That infomation is provided in the owner's guide that accompanies a new LaserWriter printer.

This Preface describes the contents of this note and the visual cues and conventions used throughout. It also lists some other books to which you can refer.

What this note contains

This developer note describes the features and special capabilities of the Personal LaserWriter NTR printer controller as well as the Adobe PostScript Level 2 operators that are specific to the printer.

Here is a brief outline of the contents:

- Chapter 1, "Introduction to the Personal LaserWriter NTR Printer," describes the features of the printer including the built-in communication channels of the Personal LaserWriter NTR printer and features of the Hewlett-Packard LaserJet IIP emulation mode.
- Chapter 2, "PostScript Level 2 for the Personal LaserWriter NTR Printer," describes the PostScript Level 2 operators that are not documented elsewhere and documented operators that produce unique results on the Personal LaserWriter NTR printer.
- Chapter 3, "TrueType Fonts," describes the TrueType downloadable PostScript font format for LaserWriter printers.

Conventions used in this note

Look for these visual cues throughout this note:

◆ *Note:* Notes like this contain supplementary information.

♦ **Important** Notes like this contain information that is essential. ♦

A special typeface is used for characters that you type or for lines of program code:

```
It looks like this.
```

Hexadecimal numbers are preceded by a dollar sign ($). For example, the hexadecimal equivalent of decimal 16 is written as $10.

Where to look for more information

This developer note assumes that you are familiar with printer technology and understand how to operate and program LaserWriter printers. However, additional information and ideas are presented in the following documents:

- The owner's guide that is shipped with every LaserWriter printer explains how to set up the printer in the standard configuration. The guide gives basic operating information, such as how to load toner cartridges, load the paper tray, set up the configuration switch for your communication environment, set up an external hard disk for fonts, and some basic troubleshooting information.
- The *LaserWriter Reference,* published by Addison-Wesley, describes the capabilities of the LaserWriter Plus, the LaserWriter IINT, and the LaserWriter IINTX printers. It also includes information that is not in this note about fonts and communicating with LaserWriter printers over the serial channels.
- The *PostScript Language Reference Manual,* second edition, published by Addison-Wesley, is required for anyone planning to write programs in the PostScript Level 2 programming language.
- The *PostScript Language Reference Manual Supplement,* available from Adobe Systems, Inc., is a supplement to the *PostScript Language Reference Manual,* second edition.
- The *PostScript Language Tutorial and Cookbook,* published by Addison-Wesley, provides a basic introduction to the PostScript programming language. It also includes sample PostScript programs to help you quickly understand how the PostScript programming language works.
- The *PostScript Language Program Design,* published by Addison-Wesley, is written for programmers who want to take advantage of the PostScript programming language to design efficient PostScript programs and printer drivers.

Chapter 1 Introduction to the Personal LaserWriter NTR Printer

The Personal LaserWriter NTR printer is Apple's newest entry into the LaserWriter family of printers. The Personal LaserWriter NTR printer replaces the Personal LaserWriter NT printer. The new printer has enhanced processing capabilities and in addition to the RS-422 and RS-232 serial ports has a Centronics parallel port. This note describes the features and capabilities of the Personal LaserWriter NTR printer controller.

The Personal LaserWriter NTR printer controller is plug-in compatible with the Personal LaserWriter NT printer. The Personal LaserWriter NTR printer controller can be an upgrade to the Personal LaserWriter NT printer.

The Personal LaserWriter NTR printer controller supports the Adobe™ PostScript™ Level 2 programming language. The original Personal LaserWriter NT printer supports PostScript Level 1. This fact is important if you intend to create programs that do not use the Apple LaserWriter driver software. You will need to refer to the *PostScript Language Reference Manual,* second edition, for information about compatibility with PostScript programs written for earlier versions of the PostScript interpreter and to this note for descriptions of the operators that support specific features of the Personal LaserWriter NTR printer controller.

Features of the Personal LaserWriter NTR printer

The features of the Personal LaserWriter NTR printer include

- a laser print engine with 4-page-per-minute capability
- an Advanced Micro Devices AMD 29005 reduced instruction set computing (RISC) microprocessor—16 MHz
- 34 built-in LaserWriter fonts
- a built-in TrueType font scaler
- a built-in PostScript Level 2 interpreter (backward compatibility with Level 1)
- a standard 70-page paper cassette with an optional 250-page paper cassette
- 3 MB to 4 MB of memory (printing legal-size pages requires a minimum of 4 MB of printer memory for black and white)
- interfaces for LocalTalk and RS-422 and RS-232 serial I/O
- a Centronics parallel port
- port arbitration for simultaneous communication over all I/O channels
- serial communications rates up to 57,600 baud for the RS-422 channel and 38,400 baud for the RS-232 channel
- emulation of the Hewlett-Packard LaserJet IIP printer

Enhanced processing speed

The Personal LaserWriter NTR printer uses an Advanced Micro Devices 29005 RISC microprocessor to provide up to double the computing performance of the Personal LaserWriter NT printer.

Simultaneous communication

The Personal LaserWriter NTR printer supports port arbitration, which allows simultaneous communication over all I/O channels. This feature makes it easier for the Personal LaserWriter NTR printer to work in multicomputer environments. Macintosh computers can work over the LocalTalk channels while IBM DOS or other networked machines work on the LocalTalk and RS-232/RS-422 serial channels. Individual IBM DOS machines or servers can print directly through the Centronics parallel channel. All channels can be connected and can receive input simultaneously. Print jobs are processed on a first-come, first-served basis. Each channel can operate under a different protocol, and each print job can select either the PostScript or the LaserJet IIP interpreter.

See "The Mode-Setting Switch," "Communication Channels," and "LaserJet IIP Emulation" later in this chapter, and Chapter 2, "PostScript Level 2 for the Personal LaserWriter NTR Printer," for more information about selecting communication protocols and the built-in emulator.

The Adobe PostScript Level 2 programming language

The Personal LaserWriter NTR printer uses the Adobe PostScript Level 2 programming language. That version of the PostScript language has features that are not described in Apple's *LaserWriter Reference* or in the first edition of Adobe's *PostScript Language Reference Manual.* The specific Personal LaserWriter NTR printer features included in PostScript Level 2 are described later, in Chapter 2. The complete description of the Level 2 implementation is provided in Adobe's *PostScript Language Reference Manual,* second edition.

PostScript Level 1 compatibility

PostScript Level 1 `statusdict` compatibility operators are provided in the PostScript Level 2 implementation in the Personal LaserWriter NTR printer to maintain a level of compatibility with the existing PostScript Level 1 LaserWriter driver software. The compatibility operators are listed in "PostScript Level 2 Compatibility Operators" in Chapter 2.

The LaserWriter Utility and LaserWriter driver

The LaserWriter Utility program shipped with each Personal LaserWriter NTR printer provides for user control and configuration of the printer.

The Macintosh LaserWriter driver and Printing Manager provide a general printer interface to Apple LaserWriter printers that should meet the needs of most Macintosh applications. Very few operations require bypassing the Printing Manager. If your application implements device-dependent features, it may be incompatible with future Apple printers. Let the Macintosh printer interface, driver software, and LaserWriter printer do the work for you.

Mode switch and status lights

This section describes the mode switch and the status lights on the outside of the Personal LaserWriter NTR printer.

The mode-setting switch

The Personal LaserWriter NTR printer controller has a 10-position pushwheel switch for setting the printer to work in different communication environments. The first six switch positions (0–5) provide a set of fixed parameter sets for each of the LaserWriter communication channels. Those parameter sets cannot be changed with PostScript operators, although they can be changed when the switch is set to positions 6 through 9. Table 1-1 shows the default parameter values for each switch setting on a Personal LaserWriter NTR printer.

■ **Table 1-1** Switch settings and default parameter values

Switch setting	I/O port	Default parameter values
0	8-pin	LocalTalk, PostScript batch
	25-pin	RS-232 9600, 8, 1, N, XON/XOFF, PostScript batch
	36-pin	Centronics, PostScript batch
1	8-pin	RS-422 9600, 7, 1, N, XON/XOFF, PostScript batch
	25-pin	RS-232 9600, 7, 1, N, XON/XOFF, PostScript batch
	36-pin	Centronics, PostScript batch
2	8-pin	LocalTalk, LaserJet IIP emulation
	25-pin	RS-232 9600, 8, 1, N, XON/XOFF, LaserJet IIP emulation
	36-pin	Centronics, LaserJet IIP emulation
3	8-pin	LocalTalk, LaserJet IIP emulation
	25-pin	RS-232 9600, 8, 1, N, DTR, LaserJet IIP emulation
	36-pin	Centronics, LaserJet IIP emulation
4	8-pin	RS-422 1200, 7, 1, N, XON/XOFF, PostScript batch
	25-pin	RS-232 1200, 7, 1, N, XON/XOFF, PostScript batch
	36-pin	Centronics, PostScript batch
5	8-pin	RS-422 9600, 8, 1, N, XON/XOFF, PostScript batch
	25-pin	RS-232 9600, 8, 1, N, DTR, PostScript batch
	36-pin	Centronics, PostScript batch
6	8-pin	(RS-422) (9600), (8), (1), (N), (XON/XOFF), (PostScript batch)
	25-pin	RS-232 (9600), (8), (1), (N), (XON/XOFF), (PostScript batch)
	36-pin	Centronics, (PostScript batch)
7	8-pin	(LocalTalk), (PostScript batch)
	25-pin	RS-232 (9600), (8), (1), (N), (XON/XOFF), (PostScript batch)
	36-pin	Centronics, (PostScript batch)
8	8-pin	(LocalTalk), (PostScript batch)
	25-pin	RS-232 (9600), (8), (1), (N), (XON/XOFF), (PostScript batch)
	36-pin	Centronics, (LaserJet IIP emulation)
9	8-pin	(LocalTalk), (PostScript batch)
	25-pin	RS-232 (9600), (8), (1), (N), (XON/XOFF), (PostScript binary)
	36-pin	Centronics, (PostScript batch)

Note: Parameters in parentheses can be changed by using the appropriate device parameters for the PostScript Level 2 `setdevparams` operator or by using the PostScript Level 1 `sccbatch` operator. In switch positions 6 through 9 any port can be enabled or disabled via PostScript software control.

You can get the number of the current switch position by using the PostScript Level 2 system parameter `PrinterMode` or by choosing Configure Communication in the Utilities menu of the LaserWriter Utility program. You can also look at the switch on the back of the printer.

Switch positions 0 through 5 select sets of communications parameters that are fixed. Switch positions 6 through 9 select sets of parameters that include user-settable parameters; those parameters can be changed by using the appropriate device parameter for the PostScript Level 2 `setdevparams` operator or by using the PostScript Level 1 `setsccbatch` operator. Parameter values and semantics for the PostScript Level 2 `setdevparams` operator are provided in Chapter 2, "PostScript Level 2 for the Personal LaserWriter NTR Printer." Information about the `setsccbatch` operator is provided in "Compatibility Operators for Setting Serial Communication Parameters" in Chapter 2.

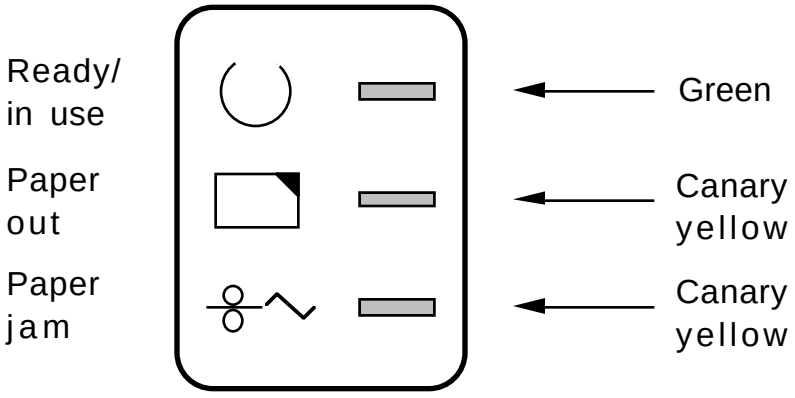
♦ **Important** Changing the mode switch setting during operation affects printing in progress immediately. The current job is aborted by the execution of a PostScript language interrupt, and the printer looks for a new job sent with the communication parameters indicated by the new switch setting. The communication parameters designated by the new switch setting become active after two seconds have elapsed. If the host computer continues to run the job that was in progress, the data it sends to the printer may cause unpredictable results. It is best to change the switch setting between jobs.

Additionally, if the switch settings are changed on a LaserWriter connected to an AppleTalk network, other users may not be aware of the new communication parameters, which may cause unpredictable results. ♦

The status lights

The Personal LaserWriter NTR printer has three colored lights that provide a simple visual indication of what the printer is doing. Figure 1-1 shows the appearance of the lights.

■ **Figure 1-1** Status lights



Different combinations of lights indicate the different states of operation, as shown in Table 1-2. The lights are also used as an indication of machine malfunctions, which are described in Table 1-3.

■ **Table 1-2** Status lights and states of operation

Operating state	Description
Busy	When the green light is flashing, the machine is busy executing a user's job. (Immediately after power has been turned on, the light flashes while the machine computes the test page to be printed.)
Idle	When the green light is on continuously, the machine is idle: it is awaiting the next user job.
Jam	When the bottom yellow light is on, a sheet of paper has failed to feed from the paper tray or has jammed inside the printer. The user can clear the jam by opening the cover of the printer and removing the paper; it is not necessary to turn the machine off before doing this.
Manual feed	When the middle yellow light is flashing, the machine is waiting for the user to feed paper into the multipurpose tray.
No paper	When the middle yellow light is on continuously, the selected paper tray is either empty or absent.

■ **Table 1-3** Status lights and malfunction conditions

Malfunction	Description
Controller	When the yellow lights blink alternately—one on, the other off, then reverse—the printer's controller needs repair.
Engine	When the yellow lights blink together—both on, then both off—the printer's print engine needs repair.

Startup page, configuration page, and printer page types

This section describes contents of the startup page and configuration page of the Personal LaserWriter NTR and defines the page types the printer supports.

Startup page

When the Personal LaserWriter NTR printer is powered on, it normally prints a startup page containing information about the printer hardware configuration. This information includes the name and model of the printer, the various communication channel parameters, the number of fonts in ROM, and the total amount of installed RAM.

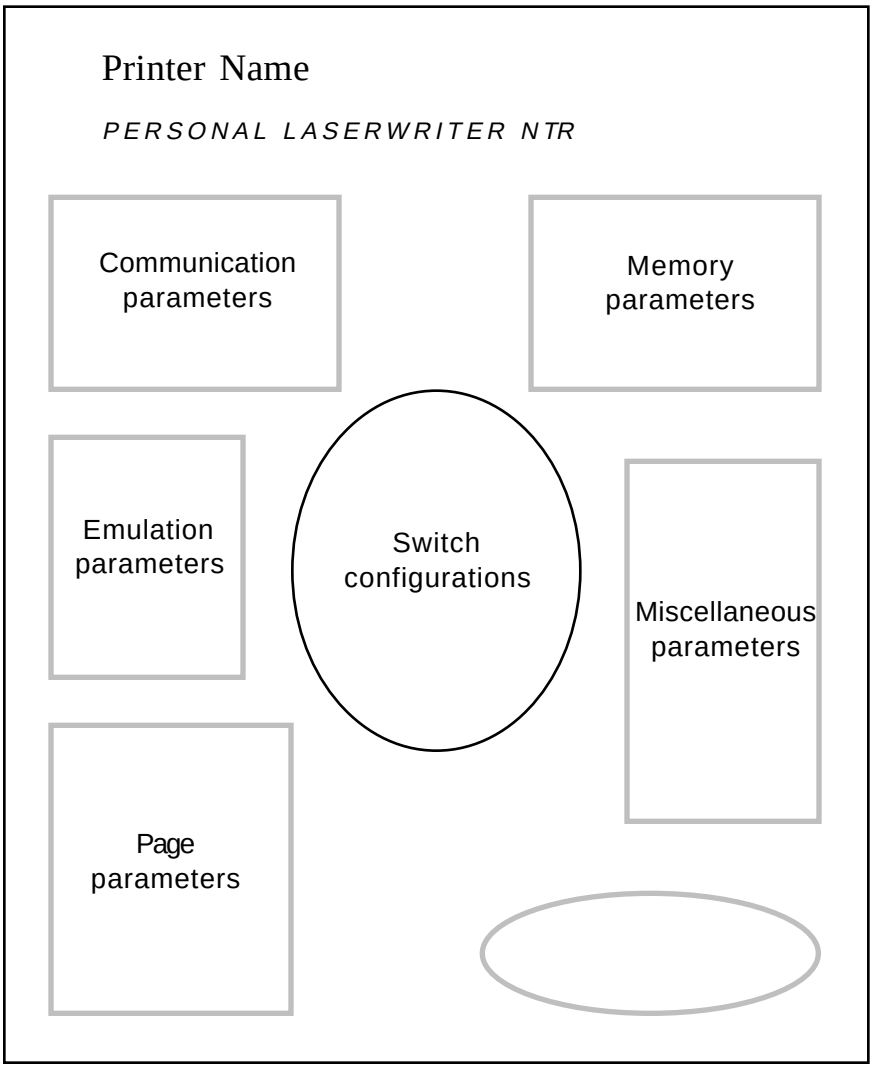
You can disable printing of the startup page by choosing Set Startup Page from the Utilities menu in the LaserWriter Utility program, by setting the `DoStartPage` system parameter of the PostScript Level 2 operator `setsystemparams` to `false`, or by setting the PostScript Level 1 compatibility operator `dostartpage` in `statusdict` to `false`.

Configuration page

The configuration page describes the current communications parameters and other values stored in the printer's persistent memory. The configuration page shows the parameters in logical groups on the page. You can print the configuration page by choosing Print Configuration Page from the Utilities menu of the LaserWriter Utility program.

Figure 1-2 shows the names of the groups and their positions on the Personal LaserWriter NTR configuration page. The following sections describe the parameters in each group.

■ **Figure 1-2** Layout of configuration page



Printer name

At the top of the test page is the printer name, as specified by the `printername` operator. Below it are the Apple logo and the model name, Personal LaserWriter NTR.

Communication parameters

The group on the left near the top shows the communication parameters for the DIN-8 port. Those parameters are

- LocalTalk type
- LocalTalk node ID

Memory parameters

The group on the right near the top shows the current memory parameters. A bar graph shows the memory allocation in five categories:

- display list
- font cache
- form cache
- pattern cache
- screen storage

This group also shows the total amount of RAM installed and the integrity of the EEROM.

Emulation parameters

The left side of the test page shows the values for three parameters. Those parameters are

- available emulators and version numbers
- startup page setting (on or off)
- printer serial number

The stored value of the serial number has the high bit set; the value shown on the test page is the corresponding positive value (with the high bit off).

Switch configurations

The center of the test page shows the parameter values for each of the different switch settings, as listed in Table 1-1.

Miscellaneous parameters

To the right of the switch settings, the test page shows the current settings of the following miscellaneous parameters:

- job timeout value, in seconds
- manual-feed timeout value, in seconds
- wait timeout value, in seconds
- system administrator password

The space for the system administrator password displays "0" (default value) if no password has been set; otherwise, it displays "Custom."

Page parameters

The lower-left portion of the test page shows the following parameters:

- default margin offsets, in points
- default matrix
- page sizes

The page sizes are the width and length, in points and inches, of the image areas of the different page types: `/a4`, `/a4small`, `/b5`, `/legal`, `/letter`, and `/lettersmall`.

The matrix values are the default values of the transformation matrix that transforms the user coordinate space to device space.

Page types

The page types for the Personal LaserWriter NTR are the same as those described in Chapter 4 of the *LaserWriter Reference.*

At the beginning of each job, the server selects the default paper tray, as assigned by the `defaultpapertray` operator. (See "Compatibility Operators for Setting System Parameters" in Chapter 2.) If the default is the main cassette, the server can detect its size and install the appropriate image region. If the default is the multipurpose tray, the server uses the image region most recently installed by means of the `setdefaultmultipurposepapertraysize` operator. When the multipurpose tray is selected in this way or with the `setpapertray` operator, it is treated just like the main cassette. Several sheets of paper may be stacked in it, and it feeds continuously until it is empty, at which time the paper-out light comes on. If a job requires a particular paper size, it should invoke one of the paper tray selection operators listed in Table 1-4 before it generates an image. That paper tray selection stays in effect for the duration of the job; the server restores the default paper tray selection when that job is finished.

■ **Table 1-4** Paper tray selection operators

Operator	Description
`a4tray`	Selects the paper tray containing A4-size paper and sets the page type to either `a4` or `a4small`, depending on the value of *pagetype*. This operator raises the PostScript language error `rangecheck` if no paper tray contains A4-size paper.
`b5tray`	Selects the paper tray containing B5-size paper and sets the page type to `b5`. This operator raises the PostScript language error `rangecheck` if no paper tray contains B5-size paper.
`legaltray`	Selects the paper tray containing legal-size paper and sets the page type to `legal`. This operator raises the PostScript language error `rangecheck` if no paper tray contains legal-size paper.
`lettertray`	Selects the paper tray containing letter-size paper and sets the page type to either `letter` or `lettersmall`, depending on the value of *pagetype*.

Communication channels

The Personal LaserWriter NTR printer has four communication channels that use three physical ports. The communication channels are LocalTalk, RS-422, RS-232, and parallel.

Serial channels

The three serial channels use two physical ports: LocalTalk and RS-422 through the mini DIN-8 port and RS-232 through the 25-pin port. The PostScript device parameter names for those two ports are `%Serial%` for the 25-pin port in RS-232 mode, `%SerialB%` for the mini DIN-8 port in RS-422 mode, and `%LocalTalk%` for the mini DIN-8 port in LocalTalk mode. The pin assignments for the mini DIN-8 port are shown in Table 1-5. The pin assignments for the 25-pin RS-232 port are shown in Table 1-6.

- **Table 1-5** Mini DIN-8 RS-422 port pin assignments

Pin number	Signal	Description
1	HSKo	Handshake output; not connected
2	HSKi	Handshake input; connected to the SCC (Serial Communications Controller) transmit/receive clock (TRxC)
3	\TxD	Transmit data –; connected to the SCC transmit data (TxD)
4	SG	Signal ground; connected to logic and chassis ground
5	\RxD	Receive data –; connected to the SCC receive rata (RxD)
6	TxD	Transmit data +; connected to the SCC transmit data (TxD)
7	None	Not connected
8	RxD	Receive data +; connected to the Scc receive data (RxD)

■ **Table 1-6** 25-pin RS-232 port pin assignments

Pin number	Signal	Description
1	Shield	Protective ground (RFI/ESD cable shield)
2	TxD	Transmit data
3	RxD	Receive data
4	RTS	Request to send (message ready)
5	CTS	Clear to send (modem operational and remote connected)
6	DSR	Data set ready (power on modem)
7	GND	Signal ground
8	DCD	Data carrier detect (received tone from remote modem)
20	DTR	Data terminal ready (ready to receive data)
22	RI	Ring indicator (telephone line is ringing)

Parallel channel

The parallel channel uses the 36-pin Centronics port. The PostScript device parameter name for the parallel port is `%Parallel%`. The pin assignments for the Centronics port are shown in Table 1-7. Descriptions of the PostScript Level 2 device parameters for controlling the communication channels are provided in "Communications" in Chapter 2.

■ **Table 1-7** Centronics port pin assignments

Pin number	Signal	Singnal status	Pin number	Signal	Singnal status
1	/Data strobe	Input	19	Signal ground	None
2	Data 1	Input	20	Signal ground	None
3	Data 2	Input	21	Signal ground	None
4	Data 3	Input	22	Signal ground	None
5	Data 4	Input	23	Signal ground	None
6	Data 5	Input	24	Signal ground	None
7	Data 6	Input	25	Signal ground	None
8	Data 7	Input	26	Signal ground	None
9	Data 8	Input	27	Signal ground	None
10	/Acknowledge	Output	28	Signal ground	None
11	Busy	Output	29	Signal ground	None
12	Paper error	Output	30	Signal ground	None
13	Select out	Output	31	/Prime	Input
14	Auto feed	Input	32	/Fault	Output
15	Select in	Input	33	None	None
16	Signal ground	None	34	None	None
17	Chassis ground	None	35	None	None
18	None	None	36	None	None

Communicating with an IBM PC via the serial channel

People using the Personal LaserWriter NTR printer with an IBM PC can set up serial port 1 on the PC for communication with the printer by issuing one of the following sets of commands. These commands set the data rate to 9600 baud and map printer output to serial port 1.

The commands to configure the PC serial port are

```
MODE COM1:9600,n,8,1
MODE LPT1:=COM1:
```

The first `MODE` command sets the data rate to 9600 baud, sets the number of data bits and stop bits to 8 and 1, respectively, and sets a long printer timeout. The long timeout allows the host to wait several seconds for DSR to be reasserted. The second `MODE` command maps the printer output to the serial port.

- *Note:* The commands by themselves are not sufficient to support XON/XOFF flow control. Some applications may handle this protocol themselves; otherwise, the user should install a different MS-DOS printer driver to avoid communication problems while printing large documents.

The commands to initiate DTR flow control are

```
MODE COM1:9600,n,8,1,p
MODE LPT1:=COM1:
```

For best results, the LaserWriter printer communication mode switch should be configured for the 9600-baud DTR setting on the 25-pin RS-232 channel, and the PC should be set for DTR flow control.

LaserJet IIP emulation

The Personal LaserWriter NTR printer has a built-in Hewlett-Packard LaserJet IIP emulator. The emulator is Hewlett-Packard PCL (printer control language) Level 4 compatible. The features of the LaserJet IIP emulator are documented in Chapter 2, "PostScript Level 2 for the Personal LaserWriter NTR Printer," and in Chapter 3 of the *LaserWriter Reference.*

Selecting emulation

The Personal LaserWriter NTR printer includes a method of invoking emulation that is not described in the *LaserWriter Reference.* That method is the use of the `emulate` operator from within a PostScript language program.

Invoking emulation by setting the server to an emulation mode, as described in the *LaserWriter Reference,* has the advantage that it makes the communication protocol the same as that of the printer being emulated. The disadvantage is that the channel must be closed and reopened when switching modes via the mode switch. When the channel is closed, all buffered data is flushed and any data sent from the host before the channel is reopened is lost. Thus, that method is not appropriate if the host computer frequently switches between sending PostScript language programs and emulation input.

Invoking an emulator from within a PostScript language program makes it possible for the host computer to switch back and forth between sending PostScript language programs and emulation input. Both types of data can be in the printer's input buffer at the same time with no data loss.

The emulator may be invoked while connected to the printer in binary mode over an asynchronous RS-232 or RS-422 serial connection or during an AppleTalk PAP (Printer Access Protocol) session. For an emulator to be invoked over a serial connection, the switch should be in position 6, 7, 8, or 9 to select the binary protocol on the 25-pin connector, or in any other position where AppleTalk is selected for the Ethernet or mini DIN-8 port. The Normal serial protocol for PostScript language programs cannot be used to process emulation input because of the special meaning of several of the control characters.

A PostScript language program can invoke an emulator by using a `statusdict` procedure called `emulate`. That procedure takes from the operand stack a file and an emulator name. The file is the input source for the emulation. The emulator name operand selects which emulation is invoked. The name must be `/LaserJetIIP`, which selects the Hewlett-Packard LaserJet IIP emulator. For compatibility with older printers the `/hpcl` name will also select the LaserJet IIP emulator. The following example selects the LaserJet emulator:

```
currentfile /LaserJetIIP statusdict /emulate get exec
```

The `emulate` procedure returns at the end of the job or when either a Control-D or an <ESC><DEL>-0 is encountered in the input source. If the emulation returns because the PostScript interpreter encounters a Control-D, that Control-D also marks the end of the job. If the emulation returns because it encounters <ESC><DEL>-0, the PostScript interpreter ejects the current page and continues executing whatever was on the execution stack before it executed the `emulate` procedure. In that case, a Control-D must still be sent to terminate the job.

Before beginning the emulation, the `emulate` procedure erases the current page and initializes the graphics state. It also clears the operand stack and the dictionary stack.

Running an emulator consumes some PostScript VM (virtual memory). If `emulate` returns normally (no interrupt), that VM is reclaimed. If emulate is called with too little VM available, a `VMerror` error occurs.

You should not call the `emulate` procedure when the printer is in interactive mode.

Emulation of the Hewlett-Packard LaserJet IIP printer

This section describes features of Hewlett-Packard LaserJet IIP emulation that are different from the LaserJet IIP when implemented on the LaserWriter. For a complete description of LaserJet Plus emulation, see "Using the Hewlett-Packard LaserJet Plus Emulator" in Chapter 3 of the *LaserWriter Reference.*

The LaserJet IIP emulator is subject to the same frame buffer limitations as a PostScript job. The LaserWriter IIf and LaserWriter IIg printers use a `lettersmall` frame buffer as the default page size when the emulator is invoked. Depending on memory in the printer, this may result in output that differs from that of an actual LaserJet IIP printer. To remedy this, precede the emulation sequence with the `letter` operator or increase the amount of memory in the printer.

Hewlett-Packard LaserJet IIP printer symbol set

The most widely used symbol set on the Hewlett-Packard LaserJet IIP printer is Roman8. Four characters in the Roman8 symbol set are not in the Adobe Standard symbol set and hence are not printed by the emulator. These four characters are

- the gray patch for rubout (decimal 127)
- the overline (decimal 176)
- the solid black square (decimal 252)
- the Italian Lira symbol (decimal 175)

◆ *Note:* These four characters have been added to the Courier typeface.

Transparent communication

The bitmap graphics operators of the Hewlett-Packard LaserJet IIP printer require that 8-bit data be transmitted to the printer. Hence, when the LaserWriter IIf or LaserWriter IIg printer is in the Hewlett-Packard LaserJet IIP printer emulation mode, it configures the input channel so that all 256 characters are transmitted uninterpreted to the emulator. That uninterpreted transmission has the effect of eliminating the Control-T status request and Control-C job interrupt commands. Nevertheless, the Control-D command specifying end-of-file is still recognized.

- *Note:* The behavior described above is different from that described on page 77 of the *LaserWriter Reference,* which states that the end-of-file is lost.

Chapter 2 PostScript Level 2 for the Personal LaserWriter NTR Printer

This chapter includes descriptions of the PostScript Level 2 page device and interpreter parameters for the Personal LaserWriter NTR printer. It also describes Level 2 resource categories for the LaserWriter printer and the PostScript Level 2 compatibility operators that make it possible for PostScript Level 1 programs to print on the Personal LaserWriter NTR printer.

The information provided in this chapter will not be sufficient for teaching you how to program the Personal LaserWriter NTR. You will need to understand the PostScript Level 2 programming language and have access to the *PostScript Language Reference Manual,* second edition, and the *PostScript Language Reference Manual Supplement* (for version 2011).

- *Note:* The *PostScript Language Reference Manual,* second edition, is abbreviated as PLRM in the tables used throughout this chapter.

Device setup

The `setpagedevice` operator is used in PostScript page descriptions to specify processing requirements and select optional printer features. The `setpagedevice` operator can also be used to specify default device setup or configuration parameters to be used when the page description doesn't specify them.

The `currentpagedevice` operator is used to query the current accumulated and adjusted state of the page device. The parameters to the `setpagedevice` operator are cumulative, in that each new call to `setpagedevice` does not reset the state in total but modifies it. In addition, on each call to `setpagedevice` the resulting accumulated page device state is processed to set up the printer to accomplish the desired results. This may result in further modification of the page device state.

For more information about how the `setpagedevice` operator is used to specify the processing requirements of a document, refer to section 4.11 of the *PostScript Language Reference Manual,* second edition.

Page device parameters

Table 2-1 lists all the page device parameters present in the Personal LaserWriter NTR printer. The semantics for all parameters appear in the *PostScript Language Reference Manual,* second edition, and the *PostScript Language Reference Manual Supplement* (for version 2011).

■ **Table 2-1** Page device parameters

Key	Type	Default	Details/permissible values
`BeginPage`	*procedure*	`{pop}`	See PLRM
`EndPage`	*procedure*	`{exch pop 2 ne}`	See PLRM
`ExitJamRecovery`	*boolean*	`true`	`True`, `false`
`HWResolution`*	*array*	`[300 300]`	Only choice

(continued)

■ **Table 2-1** Page device parameters (continued)

Key	Type	Default	Details/permissible values
`ImagingBBox`	*array* or null	`null`	See PLRM
`InputAttributes`	*dictionary*	`<<0<</PageSize [x y]>>>>`	See "InputAttributes Parameter" later in this chapter
`Install`	*procedure*	`/DefaultHalftone`	See "Install Procedure" example later in this chapter
`ManualFeed`	*boolean*	`false`	`True`, `false`
`ManualFeedTimeout`	*integer*	`60`	See PLRM
`Margins`†	*array*	`[0 0]`	See PLRM
`MediaColor`	*string* or null	`null`	See PLRM
`MediaType`	*string* or null	`null`	See PLRM
`MediaWeight`	*number* or null	`null`	See PLRM
`NumCopies`	*integer* or null	`null`	See PLRM
`OutputFaceUp`†	*boolean*	`false`	`True`, `false`
`OutputPage`	*boolean*	`true`	`True`, `false`

`PageSize`	*array*	Configuration dependent
		`[612 792] letter`
		`[612 1008] legal`
		`[595 842] a4`
		`[516 729] b5`
			See PLRM

`Policies`	*dictionary*	`<</PolicyNotFound 1`
		`/PageSize 0`
		`/PolicyReport {pop}>>`
			See PLRM

* Value is constant.

† Value is persistent across power cycles.

In some configurations the default value of the `ImagingBBox` imaging area may be set to less than the full page.

InputAttributes parameter

The *x* and *y* values of the `InputAttributes` parameter will depend upon which paper tray is installed (see Table 2-2). The multipurpose tray is always present, but if the optional cassette tray assembly is not installed, there will be no input attributes entry. If the cassette tray assembly is installed, but the tray is missing, the corresponding entry in the `InputAttributes` dictionary will be set to null. The only time this can happen for the cassette tray is when the printer is powered on and the tray is not installed.

■ **Table 2-2** Slot numbers and corresponding input sources

Slot number	Input source
0	Multipurpose tray
1	Cassette tray

If a job is sent to the printer and the tray is removed, the PostScript interpreter assumes the user will install a tray of the same size and sets the attributes accordingly. If a different tray is installed, the attributes will change to reflect the characteristics of the new tray.

There are values of matching tolerance for the `PageSize` parameter: five default user space units; landscape mode (for example, [792 612]) is also valid.

Product strings

The values assigned to the product strings associated with the Personal LaserWriter NTR are shown in Table 2-3.

■ **Table 2-3** Product string values

String name	Type	Value
`languagelevel`	*integer*	2
`product`	*string*	LaserWriter Personal NTR
`version`	*string*	2010.129

Install procedure

The following sample code illustrates a default `Install` procedure.

```
/Install {

{
      /DefaultHalftone get /Halftone findresource
}
{
pop
<<
/SpotFunction //SpotFunction
/HalftoneType 1
/Frequency 60
/Angle 45
>>
} ifelse
sethalftone

% transfer function:
{} settransfer

% stroke adjustment:
% For printers, strokeadjust should be initially disabled.
false setstrokeadjust

% color rendering:
/DefaultColorRendering /ColorRendering findresource setcolorrendering

} bind readonly def
```

Interpreter parameters

Various parameters control the operation and behavior of the PostScript interpreter. Many of them have to do with allocation of memory and other resources for specific purposes. For example, there are parameters to control the maximum amount of memory used for VM, font cache, and halftone screens.

The Personal LaserWriter NTR printer is initially configured with interpreter parameter values that are appropriate for most applications. However, a PostScript language program can alter the interpreter parameters to favor certain applications or to adapt the product to special requirements. There are three classes of interpreter parameters: user parameters, system parameters, and device parameters.

For each class there are a PostScript language operator to read the parameter values and an operator to set the parameter values. The resulting six operators are `currentuserparams`, `setuserparams`, `currentsystemparams`, `setsystemparams`, `currentdevparams`, and `setdevparams`.

The semantics for parameters appear in the *PostScript Language Reference Manual,* second edition. For more recent parameters and their semantics, see the *PostScript Language Reference Manual Supplement*.

User parameters

User parameters can be altered by any PostScript language program without special authorization. The user parameters establish temporary policies on matters such as size limits that determine whether or not an item should be cached. The `setuserparams` operator sets user parameters, and the `currentuserparams` operator reads their current values. The initial value of user parameters at the time the printer is turned on for the first time is product dependent.

Unless otherwise specified, all user parameters are subject to `save` and `restore` boundaries (`restore` resets all user parameters to their values at the time of the matching `save`).

Table 2-4 is a list of the user parameters present in the Personal LaserWriter NTR printer.

■ **Table 2-4** User parameters

Key	Type	Default	Details/permissible values
`JobName`	*string*	`()`	≤ 32 characters
`JobTimeout`	*integer*	`0`	≥ 0
`MaxDictStack`	*integer*	`530`	≥ 0
`MaxExecStack`	*integer*	`10015`	≥ 0
`MaxFontItem`	*integer*	`12500`	≥ 0
`MaxFormItem`	*integer*	`100000`	≥ 0
`MaxLocalVM`	*integer*	`2147483647`	≥ 0
`MaxOpStack`	*integer*	`100000`	≥ 0
`MaxPatternItem`	*integer*	`20000`	≥ 0
`MaxScreenItem`	*integer*	`4000`	≥ 0, initial value is 3000 bytes per MB of installed RAM, with a maximum of 12000 bytes
`MaxUPathItem`	*integer*	`5000`	≥ 0
`MinFontCompress`	*integer*	`1250`	≥ 0
`VMReclaim`	*integer*	`0`	0, –1, –2
`VMThreshold`	*integer*	`40000`	≥ 0
`WaitTimeout`	*integer*	`40`	≥ 0

System parameters

System parameters alter the overall configuration of a product. For specific parameters (noted in Table 2-5) changes will persist over power-off/power-on cycles. For others they will return to the default values whenever the printer is powered on. System parameters are set using the operator `setsystemparams` and read using the operator `currentsystemparams`. Setting system parameters requires a password. System parameters are not subject to `save` and `restore` boundaries, and their values persist across jobs.

Table 2-5 is a list of the system parameters present in the Personal LaserWriter NTR printer.

■ **Table 2-5** System parameters

Key	Type	Default	Details/permissible values
`BuildTime`	*integer*	N/A	Read-only; identifies ROM creation date.
`ByteOrder`	*boolean*	`false`	Read-only.
`CurDisplayList`	*integer*	`1020`	Read-only, amount of RAM currently occupied by the display list.
`CurFontCache`	*integer*	`144544`	Read-only, amount of RAM currently occupied by the font cache.
`CurFormCache`	*integer*	See details	Read-only, ≥ 0, amount of RAM currently occupied by the form cache.
`CurInputDevice`	*string*	()	See PLRM.
`CurOutlineCache`	*integer*	See details	Read-only, ≥ 0, amount of RAM currently occupied by the outline cache.
`CurOutputDevice`	*string*	()	See PLRM.
`CurPatternCache`	*integer*	See details	Read-only, ≥ 0, amount of RAM currently occupied by the pattern cache.
`CurScreenStorage`	*integer*	`3978`	Read-only, amount of RAM currently occupied by screen storage.
`CurSourceList`	*integer*	See details	Read-only, ≥ 0.
`CurUPathCache`	*integer*	`0`	Read-only, ≥ 0.
`DoStartPage`*	*boolean*	`true`	`true`, `false`.
`FactoryDefaults`*	*boolean*	`false`	`true`, `false`. All persistent parameters except `PageCount` and `serialnumber` will be reset.
`FatalErrorAddress`	*integer*	`0`	Hardware address of last call to fatal error handler.
`GenericResourceDir`	*string*	(Resource/)	Any valid file system prefix.
`JobTimeout`*	*integer*	`0`	≥ 0.
`MaxDisplayList`	*integer*	`167772`	≥ 0; initial value is 4% of installed RAM. This number is recomputed when the RAM configuration changes.

(continued)

■ **Table 2-5** System parameters (continued)

Key	Type	Default	Details/permissible values
MaxFontCache*	*integer*	419430	≥ 0; initial value is based on amount of RAM installed: 167,772 bytes if 4 MB RAM; otherwise, 10% of installed RAM. This number is recomputed when the RAM configuration changes.
MaxFormCache	*integer*	100000	≥ 0.
MaxOutlineCache	*integer*	65536	≥ 0.
MaxPatternCache	*integer*	100000	≥ 0.
MaxScreenStorage*	*integer*	120000	≥ 0; initial value is 30,000 bytes per MB of RAM installed. This number is recomputed when the RAM configuration changes.
MaxSourceList	*integer*	24576	≥ 0.
MaxUPathCache	*integer*	300000	≥ 0.
PageCount	*integer*	0	Read-only; ≥ 0; indicates how many pages have been successfully delivered.
PrinterMode	*integer*	0	Read-only; indicates position of mode switch. Range: 0–9.
PrinterName*	*string*	String	Any string ≤ 32 characters; : and @ not allowed. Personal LaserWriter NTR.
RamSize	*integer*	3145728	Read-only; ≥ 0; indicates amount of RAM installed.
RealFormat	*string*	(IEEE)	Read-only, IEEE.
Revision	*integer*	1	Read-only; indicates ROM revision number.
StartJobPassword*	*string*	()	No more than 32 characters or digits are allowed.
StartupMode*	*integer*	0	If 0, do nothing. If 1, then find the file Sys/Start (using SearchOrder) and execute it as an unencapsulated job.
SystemParamsPassword*	*string*	Null	No more than 32 characters or digits are allowed.
ValidNV	*boolean*	true	See PLRM.
WaitTimeout*	*integer*	40	≥ 0.

* Value is persistent across power cycles.

Device parameters

Device parameters are set using the operator `setdevparams` and read using the operator `currentdevparams`. Device parameters are the same as system parameters in that they require a password, are global to the PostScript environment, and have the same persistence characteristics. As with system parameters, some of these parameters can be stored in nonvolatile memory and persist across power-off/power-on cycles.

One property that distinguishes device parameters from both system and user parameters is that device parameters may be interdependent: the validity of a value for a given parameter might depend on the value of another parameter.

Device parameters are subdivided into sets that correspond to a particular device (`%Serial%`, `%disk2%`, `%Parallel%`, and so on). Even if two products have the same device, the parameters in the set might differ, for example, because the hardware support for that device differs.

Communications

There are four communications channels in the Personal LaserWriter NTR printer. Among them, they use three physical ports. The DIN-8 connector can be configured to use either LocalTalk or RS-422 protocols. The four channels are `%Serial%` (the 25-pin connector), `%SerialB%` (the DIN-8 connector in RS-422 mode), `%LocalTalk%` (the DIN-8 connector in LocalTalk mode), and `%Parallel%` (the Centronics connector).

For each channel, there are three related parameter sets:

- nonvolatile: The set names contain the suffix `_NV`. For example, the nonvolatile parameters associated with `%SerialB%` are in a parameter set called `%SerialB_NV%`. This set is read/write and allows you to make changes that will persist across system restarts.
- pending: The set names contain the suffix `_Pending`. This set is read-only and provides information about the settings that will be in use at the beginning of the next job on the corresponding channel, assuming no parameter changes are made before then.

- RAM: The set names have no suffix. This set is read/write and allows changes to the printer that last only until the next reboot of the printer.

Changes to the RAM set will affect the pending set but not the nonvolatile set. That is, any writes to the RAM set are write-through operations that affect the contents of the pending set. Writes to the nonvolatile set are write-through operations that affect the RAM set and also write through to the pending set, but only if the mode switch is in the position that corresponds to the nonvolatile set being changed. If the switch is in any other position, only the nonvolatile parameter set is affected.

Tables 2-6 through 2-9 list the default values of the communication parameter sets. The RAM and pending sets' values are determined by the nonvolatile sets' values. The *PostScript Language Reference Manual Supplement* describes the semantics for each parameter.

- **Table 2-6** `%Serial_NVx%` communication parameter sets

Key	Type	Default values	Details/permissible values
`Baud`	*integer*	`9600`	`300`, `600`, `1200`, `1800`, `2400`, `3600`, `4800`, `7200`, `9600`, `19200`, `57600`
`CheckParity`	*boolean*	`false`	`7`, `8`
`DataBits`	*integer*	`8`	`7`, `8`
`Enabled`	*boolean*	`true`	`true`, `false`
`FlowControl`	*name*	`/XonXoff`	`/XonXoff`, `/Dtr`, `/EtxAck`
`HasNames`	*boolean*	`false`	`true`, `false`
`Interpreter`	*name*	`/PostScript`	`/PostScript`, `/LaserJetIIP`
`On`	*boolean*	`true`	`true`, `false`
`Parity`	*name*	`/None`	`/Odd`, `/Even`, `/None`
`Protocol`	*name*	`/Normal`	`/Normal`, `/Raw`, `/Binary`
`StopBits`	*integer*	`1`	`1`, `2`
`Type`	*name*	`/Communications`	See *PostScript Language Reference Manual Supplement* (for version 2011)

■ **Table 2-7** `%SerialB_NVx%` communication parameter sets

Key	Type	Default values	Details/permissible values
`Baud`	*integer*	`9600`	`300`, `600`, `1200`, `1800`, `2400`, `3600`, `4800`, `7200`, `9600`, `19200`, `38400`
`CheckParity`	*boolean*	`false`	`7`, `8`
`DataBits`	*integer*	`8`	`7`, `8`
`Enabled`	*boolean*	`true`	`true`, `false`
`FlowControl`	*name*	`/XonXoff`	`/XonXoff`, `/Dtr`, `/EtxAck`
`HasNames`	*boolean*	`false`	`true`, `false`
`Interpreter`	*name*	`/PostScript`	`/PostScript`, `/LaserJetIIP`
`On`	*boolean*	`false`	`true`, `false`
`Parity`	*name*	`/None`	`/Odd`, `/Even`, `/None`
`Protocol`	*name*	`/Normal`	`/Normal`, `/Raw`, `/Binary`
`StopBits`	*integer*	`1`	`1`, `2`
`Type`	*name*	`/Communications`	See *PostScript Language Reference Manual Supplement* (for version 2011)

■ **Table 2-8** `%LocalTalk_NVx%` communication parameter sets

Key	Type	Default values	Details/permissible values
`Enabled`	*boolean*	`true`	`true`, `false`
`HasNames`	*boolean*	`false`	`true`, `false`
`Interpreter`	*name*	`/PostScript`	`/PostScript`, `/LaserJet IIP`
`LocalTalkType`	*string*	`LaserWriter`	`LocalTalkType` parameters refer to the same nonvolatile storage. Changing the `LocalTalk_NV` `LocalTalkType` parameter changes all `_NVx` sets' `LocalTalkType`. The `_Pending` and `RAM` sets are not affected.
`NodeID`	*integer*	`0`	`N/A`
`On`	*boolean*	`true`	`true`, `false`
`Type`	*name*	`/Communications`	See *PostScript Language Reference Manual Supplement* (for version 2011)

■ **Table 2-9** `%Parallel_NVx%` communication parameter sets

Key	Type	Default values	Details/permissible values
`Enabled`	*boolean*	`true`	`true`, `false`
`HasNames`	*boolean*	`false`	`true`, `false`
`Interpreter`	*name*	`/PostScript`	`/PostScript`, `/LaserJet IIP`
`On`	*boolean*	`true`	`true`, `false`
`Protocol`	*name*	`/Normal`	`/Normal`, `/Raw`, `/Binary`
`Type`	*name*	`/Communications`	See *PostScript Language Reference Manual Supplement* (for version 2011)

Resource categories

In Level 2, PostScript objects such as fonts, patterns, and filters can be managed as open-ended collections of resources grouped into categories. A resource is requested by the resource category and name. If the resource does not reside in VM, the resource management mechanism loads it from an external source, such as a disk, a ROM cartridge, or a network file server. Named resources are discussed in section 3.9 of the *Postscript Language Reference Manual,* second edition.

Tables 2-10, 2-11, and 2-12 list the factory-installed categories and resource instances. New resources of the regular resource categories (Table 2-10) are installed by the customer. For example, font and pattern resources can be added. The categories of implicit resources (Table 2-11) represent built-in capabilities of the interpreter. For example, the `FontType` category indicates that the interpreter understands the font formats for font types 0, 1, 3, 4, 5, and 42. Some categories are also used to define new categories (Table 2-12).

Most of the instances listed are described in the *PostScript Language Reference Manual,* second edition. The following information about instances in Table 2-10 is specific to the Personal LaserWriter NTR printer.

`ProcSet`: The `ProcSet` instance `DiagnosticProcs` contains two operators, `EngineHours` and `SendCommand`. `EngineHours` pushes an integer that estimates the number of hours the printer has been turned on since it left the factory. `SendCommand` takes an integer argument, sends it to the printer engine, and returns the integer result of that command. The commands sent with the `SendCommand` operator and the results returned by the `SendCommand` operator are unique to the device manufacturer and are not provided in this document.

The `ProcSet` instance `SamplePages` contains one operator, `StartPage`. `StartPage` takes no arguments and returns no results. It rasterizes the start page and prints it.

■ **Table 2-10** Regular resource categories

Category name	Instances
ColorRendering	DefaultColorRendering
ColorSpace	No instances defined
Encoding	ISOLatin1Encoding, StandardEncoding,
Font	AvantGarde-Book, AvantGarde-BookOblique, AvantGarde-Demi, AvantGarde-DemiOblique, Bookman-Demi, Bookman-DemiItalic, Bookman-Light, Bookman-LightItalic, Courier, Courier-Bold, Courier-BoldOblique, Courier-Oblique, Helvetica, Helvetica-Bold, Helvetica-BoldOblique, Helvetica-Narrow, Helvetica-Narrow-Bold, Helvetica-Narrow-BoldOblique, Helvetica-Narrow-Oblique, Helvetica-Oblique, NewCenturySchlbk-Bold, NewCenturySchlbk-BoldItalic, NewCenturySchlbk-Italic, NewCenturySchlbk-Roman, Palatino-Bold, Palatino-BoldItalic, Palatino-Italic, Palatino-Roman, Symbol, Times-Bold, Times-BoldItalic, Times-Italic, Times-Roman, ZapfChancery-MediumItalic, ZapfDingbats
Form	No instances defined
Halftone	DefaultHalftone
Pattern	No instances defined
ProcSet	SamplePages

The following information about instances in Table 2-11 is specific to the Personal LaserWriter NTR printer.

42: The instance 42 of the implicit resource category FontType indicates that the Personal LaserWriter NTR printer is capable of rasterizing Type 42 fonts (TrueType fonts).

■ **Table 2-11** Resources with implicit instances

Category name	Instances		
ColorRenderingType	1		
ColorSpaceFamily	CIEBasedA	CIEBasedABC	DeviceCMYK
	DeviceGray	DeviceRGB	Indexed
	Pattern	Separation	
Emulator	LaserJetIIP		
Filter	ASCII85Decode	ASCII85Encode	
	ASCIIHexDecode	ASCIIHexEncode	
	CCITTFaxDecode	CCITTFaxEncode	
	DCTDecode	DCTEncode	
	LZWDecode	LZWEncode	
	NullEncode	RunLengthDecode	
	RunLengthEncode	SubFileDecode	
FMapType	2, 3, 4, 5, 6, 7, 8		
FontType	0, 1, 3, 4, 5, 42		
FormType	1		
HalftoneType	1, 2, 3, 4, 5		
ImageType	1		
IODevice	%LocalTalk%	%LocalTalk_NV%	
	%LocalTalk_Pending%		
	%Parallel%	%Parallel_NV%	
	%Parallel_Pending%		
	%Serial%	%Serial_NV%	
	%Serial_Pending%		
	%SerialB%	%SerialB_NV%	
	%SerialB_Pending%		
PatternType	1		

■ **Table 2-12** Resources used in defining new resource categories

Category name	Instances		
Category	Category	ColorRendering	ColorRenderingType
	ColorSpace	ColorSpaceFamily	Emulator
	Encoding	Filter	FMapType
	Font	FontType	Form
	FormType	Generic	Halftone
	HalftoneType	ImageType	IODevice
	Pattern	PatternType	ProcSet
Generic	No instances defined		

Emulator parameters

The emulator parameters for the Personal LaserWriter NTR printer are shown in Table 2-13. The parameters listed apply to the Hewlett-Packard LaserJet IIP emulator.

■ **Table 2-13** Emulator parameters for the Personal LaserWriter NTR printer

Key	Type	Semantics
`FontFixed`	*boolean*	If `true`, a fixed-pitch font (such as Courier) is requested. If `false`, a proportional-spaced font (such as Times) is requested.
`FontHeight`	*real*	Specifies the desired font height in points. Note that for fixed-pitch fonts, the pitch takes precedence over the height. For example, Courier in 10 pitch and 10-point height yields 12-point height, since that is the height of 10-pitch Courier.
`FontItalic`	*boolean*	If `true`, an italic (or oblique) font is requested.
`FontPitch`	*real*	Used only if `FontFixed` is `true`. In this case, it is a real number specifying the number of characters per inch.
`FontSymbolSet`	*integer*	The mapping from 7- or 8-bit numbers to glyphs that appear on the page. The value of this parameter is the number associated with this field in a downloaded font (for example, `Roman8` = `277`).

(continued)

■ **Table 2-13** Emulator parameters for the Personal LaserWriter NTR printer (continued)

Key	Type	Semantics
`FontTypeface`	*integer*	Corresponds to the number that `LaserJetIIP` has assigned to a particular font. The emulator uses this mapping and ignores fonts with other names. **Font** — **`LaserJetIIP` value** `Courier` 3 `Helvetica` 4 `Times` 5 `Letter Gothic` 6 `Prestige Elite` 8 `Orator` 10 `Optima` 17 `Garamond` 18 `Cooper Black` 19 `CooperBlack` 19 `New Century Schoolbook` 23 `University Roman` 24
`FontWeight`	*integer*	Specifies the weight or boldness of the desired font. PostScript optionally contains a weight parameter in the `FontInfo` dictionary. The weight name is a string that maps to a font weight. The mapping is given in the following table. **Weight name** — **`LaserJetIIP` value** `/Thin` -3 `/Light` -3 `/Roman` 0 `/Medium` 0 `/Book` 0 `/Regular` 0 `/Demi` 2 `/Bold` 3 `/Heavy` 4 `/Black` 4 `/UltraBold` 4 `/ExtraBold` 4

(continued)

■ **Table 2-13** Emulator parameters for the Personal LaserWriter NTR printer (continued)

Key	Type	Semantics
`Landscape`	*boolean*	If `true`, the initial orientation of the page will be landscape instead of portrait.
`LinesPerInch`	*integer*	Specifies the default value for the vertical motion index. This determines the interline spacing (and hence the number of lines on the page).
`ManualFeed`	*boolean*	See section 4.11.3 of the *PostScript Language Reference Manual,* second edition.
`MaxLJMemory`	*integer*	There are LaserJet IIP PCL escape sequences for permanently downloading fonts and macros. With the PostScript Level 2 memory allocation flexibility, the LaserJet IIP emulator will permanently acquire memory at the expense of other PostScript needs, such as virtual memory and font cache.

PostScript Level 2 compatibility operators

The PostScript language has undergone several significant extensions. It is designed to be a universal standard for device-independent page descriptions, but each PostScript language implementation supports features and capabilities particular to that implementation. Appendix D, "Compatibility Strategies," in the *PostScript Language Reference Manual,* second edition, presents guidelines for taking advantage of language extensions while maintaining compatibility with all PostScript interpreters.

Level 1 implementations provide a collection of device control and system parameter configuration operators and procedures, most of which are defined in the dictionary `statusdict`. The contents of `statusdict` are product dependent, although an attempt has been made to maintain a consistent specification for common features. It is the dictionary for product-specific operators and other definitions.

Device control and configuration of system parameters in PostScript Level 2 are accomplished in a standard way in the language through the device setup and interpreter parameter operators. However, for compatibility with existing Level 1 PostScript language driver software, which might depend on `statusdict` operators that were often present in PostScript Level 1 products, a collection of `statusdict` operators and procedures is included in each Level 2 implementation.

Almost all of these functions are implemented as PostScript language procedures that call `setpagedevice` or appropriate Level 2 operators with appropriate arguments.

Adobe Systems recommends that you do not use the `statusdict` operators in PostScript Level 2 drivers because the presence or absence of the operators is product dependent. Instead, the appropriate Level 2 standard operators should be used.

Tables 2-14 and 2-15 list by dictionary the compatibility operators present in the Personal LaserWriter NTR printer: `statusdict` in Table 2-14 and `userdict` in Table 2-15.

■ **Table 2-14** Compatibility operators for `statusdict`

Operator name
`appletalktype`
`buildtime`
`byteorder`
`checkpassword`
`defaultmultipurposepapertraysize`
`defaultpapertray`
`defaulttimeouts`
`dostartpage`
`emulate`
`hardwareiomode`
`jobname`
`jobtimeout`
`manualfeed`
`manualfeedtimeout`
`margins`
`pagecount`
`pagestackorder`
`papersize`

(continued)

■ **Table 2-14** Compatibility operators for `statusdict` (continued)

Operator name
`papertray`
`printername`
`product`
`ramsize`
`realformat`
`revision`
`sccbatch`
`sccinteractive`
`setdefaultmultipurposepapertraysize`
`setdefaulttimeouts`
`setdostartpage`
`setdosysstart`
`sethardwareiomode`
`setjobtimeout`
`setmargins`
`setpagestackorder`
`setpapertray`
`setsccbatch`
`setsccinteractive`
`setsoftwareiomode`
`softwareiomode`
`waittimeout`

■ **Table 2-15** Compatibility operators for `userdict`

Operator name
`a4`
`a4small`
`b5`
`legal`
`letter`
`lettersmall`
`note`

Compatibility operators for setting system parameters

The following descriptions are for compatibility operators that set Level 2 system parameters. Refer to the descriptions of the system parameters provided in Table 2-5 for additional information about capabilities of the operators described here.

buildtime

Syntax `- buildtime` *int*

Definition The `buildtime` compatibility operator returns an integer with the same value as the system parameter `BuildTime`.

Error `stackoverflow`

byteorder

Syntax `- byteorder` *bool*

Definition The `byteorder` compatibility operator returns a Boolean value with the same value as the system parameter `ByteOrder`.

Error `stackoverflow`

checkpassword

Syntax *string* | *int* `checkpassword` *bool*

Definition The `checkpassword` compatibility operator checks whether *string* or *int* (*int* is converted to a *string*) is a valid password for either `SystemParamsPassword` or `StartJobPassword`. If valid, `true` is returned; otherwise, `false` is returned. If either password is not set, then `true` will be returned. A returned value of `true` indicates that *string* or *int* is a valid argument to `startjob` and `exitserver`.

Errors `stackunderflow`, `typecheck`

defaultmultipurposepapertraysize

Syntax `- defaultmultipurposepapertraysize` *name bool*

Definition This operator returns the *name* and *bool* parameters used with `setdefaultmultipurposepapertraysize` to set the default multipurpose tray size. The standard value for `defaultmultipurposepapertraysize` is `/letter true`.

Error `stackoverflow`

defaultpapertray

Syntax `- defaultpapertray` *tray*

Definition Returns the default paper tray number set by `setpapertray`.

Error `stackoverflow`

defaulttimeouts

Syntax `- defaulttimeouts` *job manualfeed wait*

Definition The `defaulttimeouts` compatibility operator returns the system parameters `JobTimeout` and `WaitTimeout` and the page device parameter `ManualFeedTimeout` for *job, wait,* and *manualfeed,* respectively.

Error `stackoverflow`

dostartpage

Syntax `- dostartpage` *bool*

Definition The `dostartpage` compatibility operator returns the value of the system parameter `DoStartPage`.

Error `stackoverflow`

dosysstart

Syntax `- dosysstart` *bool*

Definition The `dosysstart` compatibility operator returns `true` if and only if the value of the system parameter `StartupMode` is `1`.

Error `stackoverflow`

emulate

Syntax *file name* `emulate -`

Definition This operator invokes one of the emulators. The *file* parameter is used as the input source for the emulation. For Personal LaserWriter NTR printer that file should always be the file returned by the `currentfile` operator. The *name* parameter selects which emulator to invoke. The acceptable name is `/hpcl`.

If a serial input channel is used as the emulation source, then the binary protocol must be selected; otherwise, a `rangecheck` error occurs.

This procedure consumes PostScript VM but restores it before returning. If too little VM is available when the procedure is invoked, a VM error occurs.

This procedure enables host computers to switch between PostScript language interpretation and the emulator.

Errors `rangecheck`, `stackunderflow`, `typecheck`, `VMerror`

pagecount

Syntax `- pagecount` *int*

Definition The `pagecount` compatibility operator returns the value of the system parameter `PageCount`.

Error `stackoverflow`

papersize

Syntax `- papersize` *name bool*

Definition This operator returns the name of the operator that would select a tray containing the current paper size. For example, if the current paper size is letter size, this operator returns the value `/lettertray`. The value of *bool* is `true` if the page feeds short edge first, `false` if the page feeds long edge first. For the Personal LaserWriter NTR, the value of *bool* is always `true`. Note that executing the operator returned by `papersize` at some later time may not choose the same tray if both trays have the same size paper installed.

Error `stackoverflow`

papertray

Syntax `- papertray` *integer*

Definition This operator returns the paper tray number set by the `setpapertray` operator. The standard value for `papertray` is the value of the `defaultpapertray` operator.

Error `stackoverflow`

printername

Syntax *string* `printername` *substring*

Definition The `printername` compatibility operator stores the value of the system parameter `PrinterName` in *string* and returns a string object designating the *substring* actually used.

Errors `rangecheck`, `stackunderflow`, `typecheck`

product

Syntax `- product` *string*

Definition The `product` compatibility operator returns a *string* with the same value as the string `product` in `systemdict`.

Error `stackoverflow`

ramsize

Syntax `- ramsize` *int*

Definition The `ramsize` compatibility operator returns an integer with the same value as the system parameter `RamSize`.

Error `stackoverflow`

realformat

Syntax `- realformat` *string*

Definition The `realformat` compatibility operator returns a *string* with the same value as the system parameter `RealFormat`.

Error `stackoverflow`

revision

Syntax `- revision` *int*

Definition The `revision` compatibility operator returns an integer with the same value as the system parameter `Revision`.

Error `stackoverflow`

setdefaultmultipurposepapertraysize

Syntax *name bool* `setdefaultmultipurposepapertraysize -`

Definition Informs the interpreter of the paper size installed in the multipurpose tray. Because the printer cannot sense that information, operators that require the paper size refer to the value stored by this operator.

This operator must be executed outside the server loop.

The *name* operand is the name of one of the standard device setup procedures: `/letter`, `/legal`, `/a4`, or `/b5`. The procedures `/lettersmall` and `/a4small` are not allowed; the value of the `pagetype` operator controls whether the image area is small or not. The *bool* parameter is included for compatibility with other PostScript printers. It specifies whether the paper is to be fed long edge first or short edge first. For all paper sizes on the Personal LaserWriter NTR printer, the value of *bool* must be `true`, meaning short edge first.

Errors `invalidaccess`, `rangecheck`, `stackunderflow`, `typecheck`

setdefaulttimeouts

Syntax *job manualfeed wait* `setdefaulttimeouts -`

Definition The `setdefaulttimeouts` compatibility operator sets the system parameters `JobTimeout` and `WaitTimeout` to *job* and *wait,* respectively, and sets the `ManualFeedTimeout` page device parameter to *manualfeed*.

Errors `invalidaccess`, `rangecheck`, `stackunderflow`, `typecheck`

setdostartpage

Syntax *bool* `setdostartpage -`

Definition The `setdostartpage` compatibility operator sets the system parameter `DoStartPage` to the value of *bool*.

This operator must be executed outside the server loop.

Errors `invalidaccess`, `stackunderflow`, `typecheck`

setdosysstart

Syntax *bool* `setdosysstart -`

Definition The `setdosysstart` compatibility operator sets the system parameter `StartupMode` according to the value of *bool*. `StartupMode` is set to `1` if *bool* is `true` and set to `0` if *bool* is `false`.

Errors `invalidaccess`, `stackunderflow`, `typecheck`

setpapertray

Syntax *integer* `setpapertray -`

Definition This operator sets the paper tray from which paper will be fed and sets the image area according to the size of paper in that tray along with the value of the `pagetype` operator. The *integer* argument must be 0 or 1, corresponding to the main cassette or the multipurpose tray, respectively.

Because this operator installs a new image area, it should be invoked before any marks are placed on the current page. If this operator is executed while an outstanding printer error exists, the interpreter waits until the error has been cleared before completing execution of this operator. That is also true of the operators such as `lettertray` and `legaltray` because they execute `setpapertray`.

Errors `rangecheck`, `stackunderflow`, `typecheck`

Compatibility operators for setting page device parameters

The following descriptions are for compatibility operators that set Level 2 page device parameters. Refer to the descriptions of the page device parameters provided in Table 2-1 for additional information about capabilities of the operators described here.

margins

Syntax `- margins` *top left*

Definition The `margins` compatibility operator returns the *x* and *y* components of the page device `Margins` parameter as *left* and *top,* respectively.

Error `stackoverflow`

pagestackorder

Syntax `- pagestackorder` *bool*

Definition The `pagestackorder` compatibility operator returns the logical complement of the page device `OutputFaceUp` Boolean parameter.

Error `stackoverflow`

setmargins

Syntax *top left* `setmargins -`

Definition The `setmargins` compatibility operator sets the page device `Margins` parameter to [*left top*].

Errors `rangecheck`, `stackunderflow`, `typecheck`

setpagestackorder

Syntax *bool* `setpagestackorder -`

Definition The `setpagestackorder` compatibility operator sets the page device `OutputFaceUp` parameter to the logical complement of *bool*. For example, if *bool* is `true`, `OutputFaceUp` is set to `false`.

Errors `stackunderflow`, `typecheck`

Compatibility operators for setting user parameters

The following descriptions are for compatibility operators that set Level 2 user parameters. Refer to the descriptions of the user parameters provided in Table 2-4 for additional information about capabilities of the operators described here.

jobname

Syntax `- jobname` *string*

Definition The `jobname` compatibility operator is a string with the same value as the user parameter `JobName`. Redefining either `jobname` or the user parameter `JobName` redefines the other to the same value.

Error `stackoverflow`

jobtimeout

Syntax `- jobtimeout` *int*

Definition The `jobtimeout` compatibility operator returns the value of the user parameter `JobTimeout`.

Error `stackoverflow`

setjobtimeout

Syntax *int* `setjobtimeout -`

Definition The `setjobtimeout` compatibility operator sets the user parameter `JobTimeout` to the value of *int*.

Error `stackoverflow`

waittimeout

Syntax `- waittimeout` *int*

Definition The `waittimeout` compatibility operator is an integer with the same value as the user parameter `WaitTimeout`. Redefining either `waittimeout` or the user parameter `WaitTimeout` redefines the other to the same value.

Error `stackoverflow`

Compatibility operators for setting device parameters

The following descriptions are for compatibility operators that set Level 2 device parameters. Refer to the descriptions of the parameters provided in Table 2-1 and Tables 2-6 through 2-9 for additional information about capabilities of the operators described here.

appletalktype

Syntax `- appletalktype` *string*

Definition The `appletalktype` compatibility operator is a string with the same value as the `LocalTalkType` device parameter in the `%LocalTalk%` parameter set.

Error `stackoverflow`

hardwareiomode

Syntax `- hardwareiomode` *int*

Definition The `hardwareiomode` compatibility operator returns *int*, which indicates a current communications channel whose corresponding device parameter `Enabled` Boolean value is `true`. Because multiple channels may be enabled, the smallest such *int* is returned. The interpretation of *int* is the following:

```
0   %Serial%
1   %Parallel%
2   %LocalTalk%
3   %SerialB%
```

Error `stackoverflow`

manualfeed

Syntax `- manualfeed` *bool*

Definition The `manualfeed` compatibility operator is a Boolean that works in conjunction with the page device parameter `ManualFeed` to determine whether a page is fed manually. If either `manualfeed` or `ManualFeed` is `true` at the time of a `showpage` or `copypage`, then that page will be fed manually; otherwise, the page will not be fed manually. The `manualfeed` compatibility operator is present in `statusdict` if and only if the page device parameter `ManualFeed` is defined for the product. The initial value of `manualfeed` at power on is `false`.

Error `stackoverflow`

manualfeedtimeout

Syntax `- manualfeedtimeout` *int*

Definition The `manualfeedtimeout` compatibility operator is an integer that works in conjunction with the page device parameter `ManualFeedTimeout` to determine the manual feed timeout for any given page. By default, `manualfeedtimeout` is not defined in `statusdict`, and in that case the value of the page device parameter `ManualFeedTimeout` is used to determine the timeout value. If a job has defined `manualfeedtimeout` to be an integer value in `statusdict`, then this value will be used instead of `ManualFeedTimeout` for the timeout value.

Error `stackoverflow`

sethardwareiomode

Syntax *int* `sethardwareiomode -`

Definition The `sethardwareiomode` compatibility operator opens specified channel(s) for communications and closes all other channels. The variable *int* specifies which communication channel(s) should be opened by setting the `On` and `Enabled` device parameters to `true`. All other channels will be explicitly closed by setting the `On` and `Enabled` parameters to `false`. The interpretation of *int* is the following:

- `0` Open `%Serial%` and `%SerialB%`. Close all others.
- `1` Open `%Parallel%`. Close all others.
- `2` Open `%LocalTalk%`. Close all others.
- `3` Open `%Serial%` and `%SerialB%`. Close all others.

Errors `invalidaccess`, `rangecheck`, `stackunderflow`, `typecheck`

setsoftwareiomode

Syntax *int* `setsoftwareiomode -`

Definition The `setsoftwareiomode` compatibility operator sets the values of the interpreter and, if appropriate, the `Protocol` device parameters for the current communications device parameter set. The meaning of *int* is the following:

`0` PostScript Normal

`1` Not defined

`2` Not defined

`4` Not defined

`5` LaserJet IIP Raw

`100` PostScript Binary

Errors `invalidaccess`, `rangecheck`, `stackunderflow`, `typecheck`

softwareiomode

Syntax `- softwareiomode` *int*

Definition The `softwareiomode` compatibility operator returns *int,* which indicates (see `setsoftwareiomode`) the interpretation mode for the current communications device.

Error `stackoverflow`

Compatibility operators for setting serial communication parameters

The following descriptions are for compatibility operators that set Level 2 serial communication parameters. Refer to the descriptions of the communication parameters provided in Tables 2-6 and 2-7 for additional information about capabilities of the operators described in this section.

The SCC (Serial Communications Controller) operators use a 1-byte options argument (an integer parameter with values in the range 0–255) that holds an encoding of four SCC parameters: stop bits, data bits, flow control, and parity. The byte is encoded as follows: bit positions 7–0, with 7 the high bit and 0 the low bit. The bit values for the options byte are shown in Tables 2-16 through 2-19.

■ **Table 2-16** Stop-bit values for the SCC compatibility operators options byte

Bit value position 7	Stop bits
0	1 stop bit
1	2 stop bits

■ **Table 2-17** Data-bit values for the SCC compatibility operators options byte

Bit positions 6 and 5	Data bits
0	Standard
1	7 bits
2	8 bits

■ **Table 2-18** Flow-control bit values for the SCC compatibility operators options byte

Bit positions 4, 3, and 2	Flow control
0	XON/XOFF
1	DTR
2	ETX/ACK

■ **Table 2-19** Parity-bit values for the SCC compatibility operators options byte

Bit positions 1 and 0	Parity
0	Space
1	Odd
2	Even
3	Mark

In Level 1 the data bits and parity interacted in a nonorthogonal manner to produce a table of possible choices for data and parity that included many common desired methods of sending data. The standard data-bits setting is present only for purposes of compatibility with earlier versions of the PostScript Level 1 SCC operators. In particular, a standard data-bit setting could always be achieved with either a 7-bit or an 8-bit data setting. In Level 2 there are analogous parameters to those given earlier for the `%Serial%` and `%SerialB%` device parameter sets. The mapping between Level 1 stop bits and flow control and Level 2 `StopBits` and `FlowControl`, respectively, is straightforward and obvious. It is not possible to provide such a one-to-one correspondence between the Level 1 notion of data bits and parity and the Level 2 parameters `DataBits` and `Parity`. Tables 2-20 and 2-21 show the conversions between PostScript Level 1 data bits and parity and Level 2 `DataBits` and `Parity`. Notice that in going from `DataBits` and `Parity` to data bits and parity, standard parity is never used as it was in Level 1.

■ **Table 2-20** Options byte–to–device parameters conversion

Data bits and parity	**`DataBits` and `Parity`**
Standard space	7 bits Space
Standard mark	8 bits None
Standard odd	7 bits Odd
Standard even	7 bits Even
7 bits space	7 bits Space
7 bits mark	7 bits Mark
7 bits odd	7 bits Odd
7 bits even	7 bits Even
8 bits space	8 bits None
8 bits mark	8 bits None
8 bits odd	8 bits Odd
8 bits even	8 bits Even

■ **Table 2-21** Device parameters–to–options byte conversion

`DataBits` and `Parity`	Data bits and parity
7 bits None	7 bits mark
7 bits Space	7 bits space
7 bits Mark	7 bits mark
7 bits Odd	7 bits odd
7 bits Even	7 bits even
8 bits None	8 bits mark
8 bits Space	8 bits space
8 bits Mark	8 bits mark
8 bits Odd	8 bits odd
8 bits Even	8 bits even

Tables 2-20 and 2-21 are defined to provide the best compatibility with PostScript Level 1 behavior. In several cases no correct choice is possible. For example, in Level 1 there was no support for 7 data bits with no parity (that is, the total number of data and parity bits is 7). The Level 2 setting of 7 bits None is imperfectly mapped to 7 bits mark. Most serial hardware does not support 8-bit Mark or 8-bit Space, and for this reason these values are never generated in mapping from Level 1 to Level 2. In fact, in Level 1 8 bits mark and 8 bits space actually provided the equivalent of the Level 2 8 bits None functionality.

sccbatch

Syntax *channel* `sccbatch` *baud options*

Definition The `sccbatch` compatibility operator returns the serial communications device parameter settings. The values are from either the `%SerialB_NV%` (if *channel* equals `9`) or the `%Serial_NV%` (if *channel* equals `25`) parameter set. The*options* parameter is encoded as described for `setsccbatch`, and the values for data bits and parity are determined by the bit positions defined in Tables 2-20 and 2-21. Baud, stop bits, and flow control are determined from the corresponding settings for the `Baud`, `StopBits`, and `FlowControl` device parameters, respectively.

Errors `rangecheck`, `stackoverflow`, `stackunderflow`, `typecheck`

sccinteractive

Syntax *channel* `sccinteractive` *baud options*

Definition The `sccinteractive` compatibility operator pops the input argument off the stack and pushes `0 0` on the stack. This operator is essentially nonoperative.

Errors `rangecheck`, `stackoverflow`, `stackunderflow`, `typecheck`

setsccbatch

Syntax *channel baud options* `setsccbatch -`

Definition The `setsccbatch` compatibility operator sets the communication device parameters for serial communications. Either the `%SerialB_NV%` (if *channel* equals `9`) or the `%Serial_NV%` (if *channel* equals `25`) settings are affected. The following device parameters are affect by *baud* and *options:* `Baud`, `StopBits`, `DataBits`, `FlowControl`, and `Parity`. `Baud`, `Stop Bits`, and `FlowControl` are set according to the corresponding values for baud, stop bits, and flow control. `DataBits` and `Parity` are set based on the bit positions defined in Tables 2-20 and 2-21. `CheckParity` is set according to the new `Parity` setting. It is set to `true` if the setting is `Odd` or `Even`, set to `false` if the setting is `Space` or `Mark`, and not changed if the setting is `None`.

Errors `invalidaccess`, `rangecheck`, `stackunderflow`, `typecheck`

setsccinteractive

Syntax *channel baud options* `setsccinteractive -`

Definition The `setsccinteractive` compatibility operator pops the three input arguments off the stack. This operator is essentially nonoperative.

Errors `rangecheck`, `stackunderflow`, `typecheck`

Page size and paper tray compatibility operators

All the page size operators, shown in Table 2-22, are in `userdict`. Each operator requests a specific paper size. The only difference among these operations is the size of paper requested and the `ImagingBBox`. The `small` operators specify a nonnull `ImagingBBox` parameter, while the non-`small` operators specify a null `ImagingBBox` parameter. These operators use the specified size as indicated in Table 2-22 as a page device `PageSize` parameter. In addition, all of these operators set `PageSize Policy` to `7`, which guarantees that the imaging area established is correct for the requested size regardless of which tray is chosen. The only error that is generated is a `limitcheck` error caused by insufficient memory for the requested imaging area. In Table 2-22 points are used as the units for the `PageSize` and `ImagingBBox` parameters (1 point is 1/72 inch).

■ **Table 2-22** Paper size compatibility operators

Operator	`PageSize`	`ImagingBBox`
`a4`	[595 842]	Null
`a4small`	[595 842]	[25 25 570 817]
`b5`	[516 729]	Null
`legal`	[612 1008]	Null
`letter`	[612 792]	Null
`lettersmall`	[612 792]	[25 25 587 767]

The `note` operator modifies the current page device settings by establishing an `ImagingBBox` parameter of [25 25 *width* minus 25 *height* minus 25] if the current `PageSize` parameter is [*width height*]. (See the `ImagingBBox` values for `lettersmall` and `a4small` in Table 2-22.)

All the paper tray operators, shown in Table 2-23, are in `statusdict`. Each operator requests a tray containing a specific paper size. The only difference among these operations is the size of paper requested. The `PageSize` parameter requested is the same as for the corresponding page size operator, and the `ImagingBBox` parameter requested is always null. These operators use the specified size as a page device `PageSize` parameter. In addition, all of these operators set the `PageSize Policy` parameter to `0`, which guarantees that a `rangecheck` error is generated if a tray containing the requested paper size is not found. In addition to the `rangecheck` error generated because the requested tray is not present, a `limitcheck` error can occur because of insufficient memory for the requested imaging area.

■ **Table 2-23** Paper tray compatibility operators

Operator	PageSize	ImagingBBox
`a4tray`	[595 842]	Null
`b5tray`	[516 729]	Null
`legaltray`	[612 1008]	Null
`lettertray`	[612 792]	Null

Chapter 3 **TrueType Fonts**

The Personal LaserWriter NTR printer has the TrueType font-scaling software built in. This chapter describes the behavior of the TrueType downloadable PostScript font format as it applies to the Personal LaserWriter NTR printer, which has the built-in TrueType font scaler, and LaserWriter printers and other PostScript devices without the built-in TrueType font scaler.

PostScript devices are divided into three classes in this chapter. The Personal LaserWriter NTR printer is a Class A device, as described in this chapter. Additional information is included about the Class B and Class C devices, which do not have the built-in TrueType font scaler, to give a complete overview of the behavior of TrueType with existing printers in the LaserWriter printer family and other PostScript devices.

- *Note:* This chapter uses the term *PostScript device*. Generally, you can substitute *printer* for *device*, since this note is about LaserWriter printers.

The TrueType font format is designed to be as universally standardized as possible, but it is constrained to the existence of PostScript implementations in older printers and the restrictions that those implementations impose. The TrueType format, even within the constraints, is very efficient and of very high quality. To support current users, the TrueType implementation is designed to run efficiently on the large installed base of LaserWriter printers. Future printers may be optimized to take advantage of any enhancements to TrueType, and the format is designed to allow these enhancements to be incorporated easily and dynamically.

The TrueType format places all PostScript devices into one of three class categories. The classes are the following:

- Class A devices are those with a TrueType scaler embedded in the PostScript device. (The Personal LaserWriter NTR printer is a Class A device.)
- Class B devices are those with TrueType font-scaling code downloaded to the device separately from the font itself.
- Class C devices do not have TrueType available in any form for example, a third–party PostScript compatible printer.

The downloadable TrueType font format is guaranteed to be usable on any class device (with one exception, Class C devices), albeit with widely varying performance and quality characteristics. The font format as described in this note is designed for optimum performance in each class of device, and no class pays a performance or quality penalty because of optimizations in another. Class C devices will exhibit performance and quality degradations due to their inherent inability to use the TrueType enhancements. All third-party 68000-based Adobe PostScript printers, because they can handle downloaded assembly code, are treated as Class B devices.

TrueType font format

The TrueType PostScript format incorporates three component pieces: the TrueType code, a small set of procedures defined for use only during TrueType processing (hereafter referred to as the patch), and the font definition.

TrueType code

The first component, the TrueType code, is the TrueType font-scaling code. It is partitioned into three pieces and sent to the device on demand for each document that uses an `'sfnt'` resource. This code is encrypted using the Adobe encryption mechanism and depends on the existence of the `eexec` and `cexec` operators. Because of the code encryption method, the downloadable TrueType code is usable only on Class B devices and is discarded on all other class devices. It is important to note that if `eexec` and `cexec` are defined on a printer, their implementation must be Adobe PostScript compatible. Also, since the TrueType code is encrypted 68000-family code, it will not run on a printer that is not based on the 68000 family of processors. In the case of the LaserWriter 7.0 driver, the amount of available VM on the printer when the driver first encounters an `'sfnt'` resource in a document is also a factor for deciding whether the TrueType code is downloaded. If the printer has less than 120,000 bytes of VM available, the TrueType code will not be downloaded.

In the LaserWriter 7.0 driver, the TrueType code is partitioned into three physical pieces because of its relatively large size. When this code is downloaded, four new operators are defined in PostScript. A PostScript dictionary, called `TrueDict`, is created in which those four operators (as well as some version information) are stored. One operator is used to initialize TrueType for each new `'sfnt'` resource, and the others are used within the `BuildChar` procedure in the font.

The TrueType code renders characters in either bitmap or PostScript path form. The path form is invoked only when a character path is required during rendering via `charpath` or an outline (PaintType 2) style. If the bitmap size for a character exceeds 10,000 bytes (which is roughly the memory needed for a 100-point character at 300 dpi), the scaler is asked to band the bitmap, and the character is printed in bands. Future drivers or other applications may download different operators. If so, these operators will have different names if their semantics differ from those defined by the 7.0 driver. The entries in the font dictionary for a Class B printer (defined in “TrueType Font Dictionary Entries,” later in this chapter) will remain the same.

Patch

The second component is a patch to redefine the PostScript `charpath` operator. This patch is used to signal whether characters are to be rendered using the PostScript path or via a bitmap. The patched `charpath` operator simply sets a global flag to signal that `charpath` is in effect. This flag is then examined when characters are being rendered. If the flag is `true`, the characters are always constructed using a PostScript path rather than a bitmap. As with the TrueType code, the redefinition of PostScript operators is ignored on Class A and Class C devices.

The TrueType font

The third component is the actual TrueType font. The font has the minimum but essential parts of a normal PostScript font, namely, a font dictionary containing a font type, a font matrix, a font bounding box, and an encoding vector. In addition, it should contain a font name, a paint type, a stroke width (for outline styles), TrueType `'sfnt'` font data as it exists in the `'sfnt'` resource on the host, and, for Class B devices only, the TrueType state information and a `BuildChar` procedure. The bulk of the font is, of course, the `'sfnt'` font data.

The `FontType` entry for a TrueType font is `42` for Class A devices, and the `BuildChar` procedure is therefore implicit. For example, based on the `FontType` entry, the font-rendering machinery will know where to find and how to execute the font data. (The Type 42 font format is described in "TrueType Font Dictionary Entries," later in this chapter.) For Class B devices, the `FontType` entry is `3`, indicating that it is a user-defined font as understood by PostScript. For Class C devices, the `FontType` entry is `1`.

The TrueType font also has a `UniqueID` entry that is a 24-bit number derived from the checksum in the `'sfnt'` header. The presence of `UniqueID` in the font makes the PostScript font cache operate more efficiently and avoids rerendering characters across jobs. There are two `UniqueID` entries given to the font, one for the hinted font (Class A or Class B) and the other for the unhinted font (Class C). The `UniqueID` entry for a Class C font is further restricted to be between `4000000` and `4999999` (which is reserved as an open range by the Type 1 specification). Obviously, only one of these IDs is used on a particular printer. The reason for giving two different IDs is to avoid a situation where even though the printer is capable of rendering hinted characters, it gets unhinted characters because that's what was cached in the font cache by a previous job. (This could happen if the previous job didn't have enough memory to download TrueType code.)

The `'sfnt'` array can contain any number of data strings, no single one of which can exceed 65,536 bytes in size. The `'sfnt'` array is divided into the required number of pieces at arbitrary table or glyph boundaries within the `'sfnt'` array. (To guarantee word alignment of the data, there is always 1 extra byte at the end of each string in the `'sfnt'` array.) These strings are internally linked or combined at run time to simulate a continuous string of data. There is no loss in performance speed for this rather obtuse restriction in string size. The `'sfnt'` data exists in two forms: the actual `'sfnt'` data straight out of the `'sfnt'` resource and the unfolded glyph data (as Type 1 `CharStrings`) for Class C devices. The printer ignores the data it doesn't need so that the font size as it is stored on the printer is not increased.

The entries in a TrueType font dictionary for a Class A or Class B printer are listed and described in "TrueType Font Dictionary Entries," later in this chapter. The font dictionary for a Class C printer follows the Adobe Systems Incorporated Type 1 font format specification.

Device operation

This section describes how TrueType fonts are handled on Class A, Class B, and Class C devices.

Class A devices

On Class A devices such as the Personal LaserWriter NTR printer, the downloadable TrueType code is extraneous data and is discarded when the printer determines it is not needed. The low-level patches are similarly discarded when they are encountered. A system-level operator or flag on Class A devices is invoked to determine whether the TrueType code and patches are needed. The entries and behavior of Type 42 fonts are intended to be very similar to those of the LaserWriter built-in PostScript fonts (Type 1). Like Type 1 fonts, Type 42 fonts have an implicit `BuildChar` procedure, as opposed to the explicit `BuildChar` entry for Type 3 fonts.

When a character bitmap is needed from a Type 42 font, the character cache is checked first. If the bitmap is not cached, the character code is used as an index into the font's encoding array, returning a character name. This name is used as an index into the `CharStrings` dictionary (which is a required entry in the font dictionary). The value corresponding to the character name is an integer, representing the glyph index in the `'sfnt'` array. (The `'sfnt'` array has a table for mapping character codes to glyph indexes, but PostScript allows an extra level of indirection in this mapping to reencode fonts.) The glyph index and the `'sfnt'` data itself (from the `sfnts` entry) are used to rasterize the character.

Adobe built-in font formats (Type 1) have a capability, called *charstring procedures,* that allows user-defined characters to be added to the `CharStrings` dictionary. If the value of the `CharStrings` entry corresponding to a character name is an executable array (procedure), the following steps take place:

1. The `systemdict` dictionary and the font dictionary are pushed onto the dictionary stack.
2. The character code is pushed onto the operand stack.
3. The procedure is executed.
4. The `systemdict` dictionary and the font dictionary are popped from the dictionary stack.

The semantics of the procedure are almost identical to those of the Type 3 font format `BuildChar` procedure, except that in the Type 3 procedure, nothing is pushed onto the dictionary stack and the font dictionary is passed on the operand stack. The contents of the procedure must follow the same rules as does the Type 3 font format `BuildChar` procedure with respect to `setcharwidth`, `setcachedevice`, and so on. This behavior has existed in all PostScript font formats, but it has only been documented as part of Level 2. This behavior is part of Type 42 `BuildChar`.

Class B devices

Class B devices provide the primary motivation and design center for the TrueType font format in its current configuration. The small low-level patches are downloaded to Class B devices to assist the TrueType code in its operation and to provide the necessary hooks into the PostScript code. The definitions provided here are downloaded in the `userdict` at the beginning of every job.

If the `eexec` and `cexec` operators are provided when the TrueType code is downloaded, their implementation must be compatible with Adobe PostScript, or TrueType will not print on that implementation. (Depending on the level of compatibility, a PostScript error may be raised, or the printer may crash.)

Since the content of the font on a Class A device may differ from that on a Class B (or Class C) device, executing a PostScript `forall` operation within the context of a TrueType font dictionary will probably produce different results on different machines. This peculiarity should not be of significant concern, since the main contents and required definitions are the same.

Class C devices

One crucial assumption made by the TrueType code is that Class C devices all support the Adobe Type 1 font format. TrueType cannot be printed on a PostScript-compatible printer that cannot interpret the Type 1 font format. An alternative solution that would allow TrueType characters to be printed on any PostScript-compatible device would be to download a Type 3 (user-defined) font with a `BuildChar` procedure, which would convert TrueType data into cubic Bézier curves to be filled by PostScript. This solution, which is not documented in this note, is likely to be very inefficient. With Adobe Systems Incorporated making the Type 1 font format public, more PostScript-compatible printers will probably support the Type 1 format and therefore print TrueType.

Downloading TrueType fonts to disk

TrueType fonts may be downloaded to printers equipped with hard disk drives for storage of fonts. The entire font may be stored and used just like any other PostScript font, or, depending on the intelligence of the font-downloading utility, the font can be stripped of unnecessary items that will not be used on a particular class of printer. To facilitate the operation of intelligent font downloaders, there are several conventions that must be used for the textual definition of the font.

The first line in the PostScript font file is

`%!PS-TrueTypeFont-`*`sfntFormat-fontRevision-commentFormat`*

where *sfntFormat* is the version number of the `'sfnt'` format (from the `'sfnt'` header), *fontRevision* is the font manufacturer's revision of the font (also from the `'sfnt'` header), and *commentFormat* refers to this version of the commenting convention. An intelligent downloading utility can use this line to quickly identify TrueType fonts on a printer's hard disk.

If this line appears as the first line of a font program, the following conventions must be strictly followed or an error may occur.

- The token `/sfnts` must be followed by the token `[` and either `<` or `(`, depending on the encoding of the binary `'sfnt'` data. There may be white space, control characters (<CR>, >LF>, <TAB>), or a combination of both white space and control charcaters between these tokens.
- All of the strings defined in the `'sfnt'` array of a font program must use the same encoding (ASCII or ASCIIHex). Different font programs may use different encodings.
- The characters representing the `'sfnt'` data must follow a sequence of *N* characters of data followed by *M* characters of white space, repeating until the string's data is exhausted. The last sequence of character data may be less than *N* characters long. The last tokens in each string should be *M* characters of white space, followed by the character(s) for 1 pad byte of data, followed immediately by the string terminator (either `>` or `)`, depending on the data encoding). There may be white space, control characters, or a combination of both white space and control characters between string definitions. The values of *M* and *N* must be constants for a font program. Different font programs may use different values of *M* and *N*. The value for *N* must be between 0 and 2048, inclusive. (Note that 1024 bytes of binary `'sfnt'` data require 2048 characters to represent in the ASCIIHex encoding.)
- The last string definition in the `'sfnt'` array must be followed by the token `]` and the token `def`. There may be white space, control characters, or a combination of both white space and control characters between these tokens.

The remainder of the commenting convention involves bracketing the PostScript code for different classes of printers with `begin` and `end` comments. The `end` comments are already used by the `checkload` and `fcheckload` procedures when discarding sections of PostScript code that are not appropriate for a given class of printer. The `begin` comments are for the font downloader, which does not have a PostScript interpreter to do the discarding automatically. Because of a limitation in the `readline` operator in early versions of the PostScript interpreter, the `end` comments must be bracketed by only linefeed (ASCII 10) characters.

The following are comments that delineate sections of code and the classes of printers for which they are required. (The <SP> indicates the space character. You must enter a space at the beginning of each `end` comment line.)

`%beginsfnt`

<SP>`%endsfnt`

This brackets the creation of the common entries in the font dictionary for Class A and Class B devices. It may be discarded on Class C devices.

`%beginsfntBC`

<SP>`%endsfntBC`

This brackets the definition of the Class B–specific entries (`TrueState` and `BuildChar`) in the TrueType font dictionary. It may be discarded on Class A and Class C devices.

`%beginsfntdef`

<SP>`%endsfntdef`

This brackets the call to `definefont`, which registers the font dictionary for Class A and Class B devices. It may be discarded on Class C devices.

`%beginType1`

<SP>`%endType1`

This brackets the definition of the Type 1 font dictionary for Class C devices. It may be discarded on Class A and Class B devices.

TrueType font dictionary entries

In the Macintosh system software, TrueType fonts are represented as a resource, called `'sfnt'` for scalable font. In PostScript interpreters, fonts are represented as dictionaries with certain special key-value pairs. One of these entries, `FontType`, identifies the font format and tells the PostScript font mechanism how to interpret the remaining entries. The `FontType` entry for TrueType fonts on Class A devices is `42`. This section describes the remaining entries and their semantics for Type 42 font dictionaries.

The following tables represent the possible entries in a TrueType font dictionary for Class A or Class B devices. (Class C devices use the Type 1 font format, as documented by the *PostScript Language Reference Manual,* second edition.) Certain entries are required only for either Class A or Class B devices. Others have different values, depending on the type of device. Still others are optional and are not used by the font-rendering code itself.

A valid Type 42 font dictionary must have certain key-value pairs. Table 3-1 lists the entries common to all PostScript fonts. (Table 3-1 is similar to Tables 5.1 and 5.2 in the *PostScript Language Reference Manual,* second edition.) Table 3-2 lists entries specific to Type 1 (similar to Table 5.3 in the *PostScript Language Reference Manual,* second edition). Some of these are supported exactly as in Type 1, and others are ignored by Type 42 fonts. Table 3-3 lists the entries specific to Type 42 font dictionaries. Table 3-4 lists the entries in the optional `FontInfo` dictionary and where corresponding information is found in the `'sfnt'` format (similar to Table 5.4 in the *PostScript Language Reference Manual,* second edition).

■ **Table 3-1** Type 42 key-value pairs common to all PostScript font dictionaries

Key	Type	Semantics
`FontType`	*integer*	Required. Indicates where the information for the character descriptions is found and how it is represented. Value for TrueType fonts: `42`.
`FontMatrix`	*array*	Required. Transformation matrix for transforming the character coordinate system into the user coordinate system. TrueType fonts maintain this value internally (for example, Apple TrueType fonts use a 2048-unit coordinate system), so the PostScript coordinate system transformation is the identity matrix. Value for Type 42 fonts: `[100100]`. *Note:* Certain PostScript programs (for example, program 16 in the *PostScript Language Tutorial and Cookbook*) incorrectly assume that all PostScript fonts have a 1000-unit coordinate system. These programs may exhibit incorrect behavior when used with Type 42 fonts.

(continued)

■ **Table 3-1** Type 42 key-value pairs common to all PostScript font dictionaries (continued)

Key	Type	Semantics
`FontName`	*name*	Optional. The font name. This entry is for information only; it is not used by the PostScript interpreter. Conventional value: PostScript name from the `'sfnt'` name table.
`FontInfo`	*dictionary*	Optional. This entry is for information only; it is not used by the PostScript interpreter. See Table 3-4 for the entries that can be in this dictionary.
`LanguageLevel`	*integer*	Optional. Minimum language level required for correct behavior of the font. This entry is for information only; it is not used by the PostScript interpreter. Default value: `1`.
`WMode`	*integer*	Optional. Indicates which of two sets of metrics is used when characters are shown from this font. If this entry (or the `WMode` entry of the root font from which this font is a descendant) has the value `1`, then this font must have a `CDevProc` entry (see Table 3-2). See section 5.9 of the *PostScript Language Reference Manual,* second edition, for information about composite fonts. Default value: `0`.
`Encoding`	*array*	Required. Array of 256 names that maps character codes (integers) to character names. Note that Apple TrueType fonts have an encoding vector different from the `StandardEncoding` used by Type 1 fonts. Conventional value: derived from information in the `'sfnt'` post table.
`FontBBox`	*array*	Required. Array of four numbers in the character coordinate system giving lower-left *x*, lower-left *y*, upper-right *x*, and upper-right *y* of the font bounding box. *Note:* For compatibility with certain versions of the LaserWriter driver, this array should have the executable attribute.
`UniqueID`	*integer*	Optional. Integer in the range 0 to 16777215 ($2^{24} - 1$) that uniquely identifies this font for the purposes of caching character bitmaps and metrics. Conventional value: the lower 24 bits of the `'sfnt'` checksum.
`XUID`	*array*	Optional. Array of integers that uniquely identifies this font or any variant of it for the purposes of caching character bitmaps and metrics.

■ **Table 3-2** Entries for Type 1 specific font dictionaries

Key	Type	Semantics
`PaintType`	*integer*	Required. A code indicating how the characters of the font are to be painted: `0` The character outlines are filled. `2` The outlines (designed to be filled) are stroked. TrueType fonts are ordinarily created with a `PaintType` of `0`. A program desiring to convert it to a stroked outline font can copy the font dictionary, change the `PaintType` from `0` to `2`, add a `StrokeWidth` entry, and define a new font using this dictionary. *Note:* If `PaintType 0` is chosen, the TrueType scan converter is used to render the character. If `PaintType 2` is chosen, the grid-fitted TrueType outline is converted to PostScript path segments and the PostScript scan converter strokes the path.
`StrokeWidth`	*number*	Optional. Stroke width (in units of the *character* coordinate system) for *stroked* outline fonts (`PaintType 2`). This field is not initially present in filled font descriptions. It must be added when creating a stroked font from an existing font. *Note:* The caveat mentioned for `FontMatrix` in Table 3-1 most often manifests itself in this entry.
`Metrics`	*dictionary*	Ignored. Adding a `Metrics` entry will have no effect on a Type 42 font.
`Metrics2`	*dictionary*	Ignored. Adding a `Metrics2` entry will have no effect on a Type 42 font.
`CDevProc`	*procedure*	Optional. Procedure that algorithmically derives global changes to a font's metrics. If this font (or the root font for which this font is a descendant) has a `WMode` of `1`, this entry is required. See section 5.6.2 of the *PostScript Language Reference Manual,* second edition, for the semantics of this procedure.

(continued)

■ **Table 3-2** Entries for Type 1 specific font dictionaries (continued)

Key	Type	Semantics
`CharStrings`	*dictionary*	Required. Associates character names (keys) with glyph IDs (integers). These IDs are used to access data in the `'sfnt'` format. Every Type 42 font must have a `notdef` entry (usually with glyph ID `0`). The value can also be an executable PostScript procedure; see section 5.6.3 of the *PostScript Language Reference Manual,* second edition.
`Private`	*dictionary*	Ignored. Type 42 fonts do not require a `Private` dictionary.

■ **Table 3-3** Font dictionary entry specific to Type 42 fonts

Key	Type	Semantics
`sfnts`	*array*	Required. Array of PostScript string objects that contains the font description in the `'sfnt'` format. Because PostScript strings can be no more than 65,535 bytes long, `'sfnt'` descriptions that are longer than 65,535 bytes must be broken into separate strings. The `'sfnt'` data should be divided at both a longword and a table boundary. If a single table exceeds 64K bytes, then it should be divided at the nearest longword and glyph boundary. *Note:* For compatibility with certain versions of the LaserWriter driver, each string in the `'sfnt'` array must contain a single pad byte at the end.

■ **Table 3-4** Optional entries for `FontInfo` dictionary

Key	Type	Semantics
`FamilyName`	*string*	Human-readable name for a group of fonts that are stylistic variants of a single design. All fonts that are members of such a group should have exactly the same `FamilyName`. Conventional value: font family name from the `'sfnt'` name table.
`FullName`	*string*	Unique, human-readable name for an individual font. Conventional value: full font name from the `'sfnt'` name table.
`Notice`	*string*	Trademark or copyright notice, if applicable. Conventional value: copyright notice from the `'sfnt'` name table.
`Weight`	*string*	Human-readable name for the weight, or boldness, attribute of a font. Conventional value: font subfamily name from the `'sfnt'` name table.
`version`	*string*	Version number of the font program. Conventional value: version string from the `'sfnt'` name table (*not* the version entry in the `'sfnt'` post table). *Note:* This entry is incorrectly capitalized in Table 5.4 of the *PostScript Language Reference Manual,* second edition.
`ItalicAngle`	*number*	Angle in degrees counterclockwise from the vertical of the dominant vertical strokes in the font. Conventional value: italic angle from the `'sfnt'` post table.
`isFixedPitch`	*boolean*	If `true`, indicates that the font is a fixed-pitch (monospaced) font. Conventional value: `IsFixedPitch` Boolean value from the `'sfnt'` post table.
`UnderlinePosition`	*number*	Recommended distance from the base line for positioning underlining strokes. This number is the *y* coordinate (in character space) of the center of the stroke. Conventional value: underline position from the `'sfnt'` post table.
`UnderlineThickness`	*number*	Recommended stroke width for underlining, in units of the character coordinate system. Conventional value: underline thickness from the `'sfnt'` post table.

THE APPLE PUBLISHING SYSTEM

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and Microsoft Word software. Proof pages and final pages were created on Apple LaserWriter printers. Line art was created using Adobe™ Illustrator. PostScript™, the page-description language for the LaserWriter, was developed by Adobe Systems Incorporated.

Text type and display type are Apple's corporate font, a condensed version of ITC Garamond®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Apple Courier.

Writer: Steve Schwander