Contents

Chapter 1: Introduction  1

Chapter 2: Page Device Parameters  3
  2.1 Details Dictionaries  3
  2.2 Parameter Descriptions  4
    Media Selection  5
    Page Image Placement  5
    Rendering Control  6
    Page Delivery  8
    Error Handling  11
  2.3 Envelope Orientation in User Space  11

Chapter 3: Interpreter Parameters  13
  3.1 User Parameters  14
  3.2 System Parameters  15
  3.3 Device Parameters  18
    Device Parameter Sets  19
    Device Parameter Interdependencies  20
    Communications Parameter Sets  21
    Parameters Parameter Sets  44
    FileSystem Parameter Sets  60

Chapter 4: Resources  69
  4.1 Regular Resource Categories  69
    ControlLanguage  70
    PDL  70
    Localization  71
    HWOptions  73
  4.2 Implicit Resource Categories  73
Figures

2.1 Default user space for an envelope in portrait orientation 12
3.1 OSI (Open Systems Interconnection) layers 20
3.2 Relationship between network communications protocols and physical communications media 22
3.3 Relationship among the Communications parameter sets 25
3.4 Communications parameter sets using NV values 27
3.5 Communications parameter sets using "hard-wired" values 27
## Tables

2.1 Categories of additional page device parameters  
2.2 Additional page device parameter related to media selection  
2.3 Additional page device parameters related to page image placement  
2.4 Additional page device parameters related to rendering control  
2.5 Typical entries in a `DeviceRenderingInfo` dictionary  
2.6 Typical entries in a `PostRenderingEnhanceDetails` dictionary  
2.7 Additional page device parameters related to page delivery  
2.8 Typical entries in a `CollateDetails` dictionary  
2.9 Additional page device parameter related to error handling  

3.1 User parameters  
3.2 System parameters  
3.3 Device parameters common to all Communications parameter sets  
3.4 Additional device parameters specific to the `%Serial% (or `%SerialX%`) parameter set  
3.5 Additional device parameters specific to the `%Parallel% (or `%ParallelX%`) parameter set  
3.6 Additional device parameters specific to the `%ScsiComm% (or `%ScsiCommX%`) parameter set  
3.7 Additional device parameters specific to the `%LPR% (or `%LPRX%`) parameter set  
3.8 Additional device parameters specific to the `%AppSocket% (or `%AppSocketX%`) parameter set  
3.9 Additional device parameter specific to the `%RemotePrinter% (or `%RemotePrinterX%`) parameter set  
3.10 Additional device parameters specific to the `%PrintServer% (or `%PrintServerX%`) parameter set  
3.11 Additional device parameters specific to the `%LAT% (or `%LATX%`) parameter set  
3.12 Additional device parameters specific to the `%LocalTalk% (or `%LocalTalkX%`) parameter set  
3.13 Additional device parameters specific to the `%EtherTalk% (or `%EtherTalkX%`) parameter set
3.14 Additional device parameters specific to the %TokenTalk% (or %TokenTalkX%) parameter set 43
3.15 Forms of a node address 47
3.16 Device parameters in the %SNMP% (or %SNMPX%) parameter set 47
3.17 Device parameters in the %Syslog% (or %SyslogX%) parameter set 48
3.18 Device parameters in the %TCP% (or %TCPX%) parameter set 49
3.19 Device parameters in the %UDP% (or %UDPX%) parameter set 49
3.20 Device parameters in the %IP% (or %IPX%) parameter set 49
3.21 Device parameters in the %SPX% (or %SPXX%) parameter set 52
3.22 Device parameters in the %IPX% (or %IPXX%) parameter set 52
3.23 Device parameters in the %EthernetPhysical% (or %EthernetPhysicalX%) parameter set 54
3.24 Device parameters in the %TokenRingPhysical% (or %TokenRingPhysicalX%) parameter set 54
3.25 Device parameters in the %Scsi% (or %ScsiX%) parameter set 55
3.26 Device parameters in the %Ide% (or %IdeX%) parameter set 56
3.27 Device parameters in the %Engine% (or %EngineX%) parameter set 56
3.28 Device parameters in the %Console% (or %ConsoleX%) parameter set 57
3.29 Selected ISO 639 language codes 58
3.30 Selected ISO 3166 country codes 59
3.31 Device parameters in the %Calendar% (or %CalendarX%) parameter set 60
3.32 Device parameters in the %disk% parameter set 61
3.33 Device parameters in the %cartridge% (or %cartridgeX%) and %rom% (or %romX%) parameter sets 64
3.34 Device parameters in the %ram% (or %ramX%) parameter set 65
3.35 Device parameters in the %os% (or %osX%) parameter set 66

4.1 Entries in an instance of the ControlLanguage resource category 70
4.2 Entries in an instance of the PDL resource category 71
4.3 Entries in an instance of the Localization resource category 71
4.4 Implicit resources 73

5.1 Meanings of MetricsCount values 81
5.2 Additional entry in all image dictionaries 81
5.3 Additional trapping parameter 82
5.4 Format of crdInfoTag 83

C.1 Device parameters in the %PCL% parameter set 130
CHAPTER 1

Introduction

THIS DOCUMENT DESCRIBES extensions and features of the PostScript® language that are not documented in the PostScript Language Reference, Third Edition (referred to throughout this Supplement by the abbreviation PLR3). These include extensions to LanguageLevel 3 that are implemented in Adobe PostScript 3 versions 3010 and 3011, as well as some features that were too device-specific to include in the PostScript Language Reference and some facilities for maintaining backward compatibility with LanguageLevel 1 implementations.

• Chapter 2, “Page Device Parameters,” supplements Chapter 6 of the Third Edition. It describes additional page device parameters that are relatively device-specific—those that control features unique to a particular output device or limited to only a few devices.

• Chapter 3, “Interpreter Parameters,” supplements Appendix C of the Third Edition. It provides information about additional user, system, and device parameters.

• Chapter 4, “Resources,” supplements Section 3.9 of the Third Edition. It describes additional named resources that are relatively device-specific.

• Chapter 5, “Other Extensions,” provides information about miscellaneous extensions to the PostScript language that are not described in the Third Edition. These include new and extended font types, extensions to the contents of image dictionaries and trapping parameter dictionaries, and support for device color profiles as defined by the International Color Consortium (ICC).

• Chapter 6, “Compatibility,” supplements Chapter 8 of the Third Edition. It describes operators, procedures, and data values that are supported in LanguageLevel 2 and 3 implementations in order to maintain backward compatibility with LanguageLevel 1 implementations.
• Appendix A, “Changes since Earlier Versions,” lists changes made to the PostScript language since version 3010. These changes were incorporated into the 3011 release.


• Appendix C, “Emulator Parameter Sets,” describes parameter sets of type Emulator, which provide supporting information for emulating non-PostScript imaging systems within a PostScript imaging system.


The Supplement concludes with a Bibliography and an Index.

Note: If you discover any errors in or have any problems with this documentation, please e-mail us at

<http://doc_problems@adobe.com>

Please describe the error or problem as completely as possible and give us the document ID number (located at the foot of the cover page), the document title, and the page number or page range.
2.1 Details Dictionaries

As discussed in PLR3, Section 6.1.2, some page device features are controlled by two separate parameters: a boolean or integer parameter that enables or disables the feature as a whole and a details dictionary containing variables that determine its precise behavior. For example, on a device with a stapling capability, the boolean page device parameter Staple specifies whether to staple a document after printing, while the details dictionary StapleDetails might contain further information such as the number, location, and orientation of the staples.

The exact contents of the details dictionary for a given feature may vary from one device to another, depending on the specific capabilities and characteristics of the device. If no further configuration is possible beyond simply enabling or disabling the feature, only the boolean parameter will be present with no associated details dictionary. Applications should never assume the presence of a details dictionary without knowing the exact nature of the output device with which they are communicating.

Every details dictionary has a Type entry whose integer value determines the remaining structure and contents of the dictionary. The setpagedevice operator examines this entry and raises an undefined error if it is absent or a configuration error if it doesn’t match the value expected by the current page device. For a given feature, details dictionaries with the same Type value will have entries with exact-
ly the same names, formats, and meanings, even if intended for use with different output devices. When the values in a details dictionary are changed, the Type entry also determines whether the contents of the new dictionary overwrite those of the current one or are merged with it. To avoid conflicts, details dictionaries and their associated Type values should be registered with Adobe.

### 2.2 Parameter Descriptions

Table 2.1 lists the page device parameters covered in this section. Note, however, that not all parameters described here are applicable to all devices; consult the product documentation for a particular device to see exactly which parameters it supports. In the future, new parameters may be defined as needed to control new processing options or device features. Once defined for any device, a given parameter name will always be used for the same feature on any subsequent devices that support it.

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>DESCRIPTION</th>
<th>PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media selection</td>
<td>Select the appropriate type of paper or other physical medium.</td>
<td>ManualFeedTimeout</td>
</tr>
<tr>
<td>Page image placement</td>
<td>Specify how page images are to be rendered onto the physical medium.</td>
<td>Signature Details</td>
</tr>
<tr>
<td>Rendering control</td>
<td>Specify device-specific settings to control the rendering of page images onto the physical medium.</td>
<td>DeviceRenderingInfo, PostRenderingEnhance, PostRenderingEnhanceDetails, PreRenderingEnhance, PreRenderingEnhanceDetails</td>
</tr>
<tr>
<td>Page delivery</td>
<td>Control the disposition of physical output.</td>
<td>Bind, BindDetails, Booklet, BookletDetails, CollateDetails, Fold, FoldDetails, Laminate, OutputPage, Punch, PunchDetails, SlipSheet, Staple, StapleDetails, Trim</td>
</tr>
</tbody>
</table>
2.2 Media Selection

Table 2.2 describes an additional page device parameter related to media selection, supplementing those described in Section 6.2.1 of the PostScript Language Reference, Third Edition.

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManualFeedTimeout</td>
<td>integer</td>
<td>The number of seconds to wait for a page to be fed manually before generating a timeout error. A value of 0 means no timeout (infinite wait).</td>
</tr>
</tbody>
</table>

2.2.2 Page Image Placement

Table 2.3 describes additional page device parameters related to page image placement, supplementing those described in Section 6.2.3 of the PostScript Language Reference, Third Edition.

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature</td>
<td>boolean</td>
<td>A flag determining whether page images are to be placed on the output medium in the form of signatures (sets of pages arranged on the same sheet of physical medium in such a way that they will appear in the correct order and orientation when the medium is folded and cut). How this process is performed is device-dependent. On some devices, the device itself provides the memory and disk space needed to collect a set of page images and arrange them into a signature; on other devices, the interpreter may have to rearrange the pages and deliver them to the device as a single image representing the entire signature. In the latter case, the interpreter may need to execute multiple page descriptions and store the results in order to combine them into a</td>
</tr>
</tbody>
</table>
signature image. This use of `Signature` is supported by relatively few devices and is subject to resource limits in those that do support it.

**SignatureDetails** dictionary  
A dictionary containing device-specific parameters that determine how the page images are to be arranged in signatures. This parameter is ignored if the value of `Signature` is false.

### 2.2.3 Rendering Control

Some output devices need the ability to specify additional, device-specific settings to control the rendering of page images onto the physical output medium. The nature and meaning of these settings may be very different for different devices. The page device parameters described in Table 2.4 allow the specification of such device-specific rendering information.

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DeviceRenderingInfo</code></td>
<td>dictionary</td>
<td>A dictionary containing device-specific rendering parameters; see discussion below and Table 2.5.</td>
</tr>
<tr>
<td><code>PreRenderingEnhance</code></td>
<td>boolean</td>
<td>A flag that enables or disables image enhancement before the page image is rasterized in memory.</td>
</tr>
<tr>
<td><code>PreRenderingEnhanceDetails</code></td>
<td>dictionary</td>
<td>A dictionary containing device-specific parameters that control the operation of prerendering image enhancement on this device. This parameter is ignored if the value of <code>PreRenderingEnhance</code> is false.</td>
</tr>
<tr>
<td><code>PostRenderingEnhance</code></td>
<td>boolean</td>
<td>A flag that enables or disables image enhancement after the page image is rasterized in memory.</td>
</tr>
<tr>
<td><code>PostRenderingEnhanceDetails</code></td>
<td>dictionary</td>
<td>A dictionary containing device-specific parameters that control the operation of postrendering image enhancement on this device. This parameter is ignored if the value of <code>PostRenderingEnhance</code> is false.</td>
</tr>
</tbody>
</table>

`DeviceRenderingInfo` is an optional dictionary holding device-specific information to control the rendering process. Like a details dictionary (see Section 2.1, “Details Dictionaries”), the `DeviceRenderingInfo` dictionary has a `Type` entry whose integer value determines its structure and contents. If two different devices use `DeviceRenderingInfo` dictionaries with the same `Type`, then the dictionaries
will have exactly the same named entries and the form and meaning of each entry will be the same. This allows an application to manipulate the contents of the `DeviceRenderingInfo` dictionary based solely on the value of its `Type` entry, without needing to know anything further about the exact nature of the output device. Note, however, that `DeviceRenderingInfo` is not a details dictionary and is not enabled or disabled by a boolean device parameter; its contents are simply applied as needed to control the rendering process on the device.

Besides the `Type` entry, Table 2.5 shows two other typical `DeviceRenderingInfo` entries commonly used by PostScript devices. These are only examples; a given output device may include other entries in its `DeviceRenderingInfo` dictionary, and may or may not include the ones described here. See product-specific documentation for a complete description of the dictionary’s contents for any particular device.

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>integer</td>
<td><em>(Required)</em> A code that determines the format and meaning of the other entries in the dictionary.</td>
</tr>
<tr>
<td>ValuesPerColorComponent</td>
<td>integer</td>
<td>The number of possible values for each color component, or the number of gray levels on a monochrome device.</td>
</tr>
<tr>
<td>Limit</td>
<td>integer</td>
<td>The maximum colorant density to be applied to the output medium. The value is expressed as a percentage in the range 100–400, where 400 represents the maximum possible density on a device with four process colorants.</td>
</tr>
</tbody>
</table>

In addition to using `DeviceRenderingInfo` to control the rendering process itself, some devices may wish to perform special processing either before rendering (to set up conditions for the rendering process) or after it (to do image enhancement on the results). The page device parameters `PreRenderingEnhance` and `PostRenderingEnhance` enable or disable such processing; the corresponding details dictionaries `PreRenderingEnhanceDetails` and `PostRenderingEnhanceDetails` can hold additional, device-specific parameters to control it. As usual, these dictionaries must have a `Type` entry to characterize their structure and contents. The remaining entries are device-dependent, and are described in product-specific documentation; Table 2.6 shows some typical entries included in some devices' `PostRenderingEnhanceDetails` dictionaries.
### TABLE 2.6 Typical entries in a PostRenderingEnhanceDetails dictionary

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>integer</td>
<td><em>(Required)</em> A code that determines the format and meaning of the other entries in the dictionary.</td>
</tr>
<tr>
<td>PrintQuality</td>
<td>integer</td>
<td>An integer specifying the level of output quality desired. A zero value denotes the lowest available level, or “draft” quality.</td>
</tr>
<tr>
<td>TonerSaver</td>
<td>integer</td>
<td>An integer specifying the degree to which colors should be lightened to economize on the use of colorants. A zero value denotes no lightening; nonzero values reduce the tint levels of colorants applied to the output medium, causing (for example) full black to appear gray.</td>
</tr>
<tr>
<td>REValue</td>
<td>integer</td>
<td>An integer specifying the level of rendering enhancement desired. A zero value denotes the lowest available level of enhancement, typically none at all.</td>
</tr>
</tbody>
</table>

### 2.2.4 Page Delivery

Table 2.7 describes additional page device parameters related to page delivery, supplementing those described in Section 6.2.4 of the *PostScript Language Reference*, Third Edition.

### TABLE 2.7 Additional page device parameters related to page delivery

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OutputPage</td>
<td>boolean</td>
<td>A flag specifying whether to produce actual physical output. If the value of this parameter is false, all processing, including rasterization to a frame buffer, is performed as if the page were to be printed, but no physical output is produced. The time required to process a page thus includes everything except the time spent waiting for the marking engine; this is useful for measuring the cost of page execution.</td>
</tr>
<tr>
<td>CollateDetails</td>
<td>dictionary</td>
<td>A dictionary containing device-specific parameters that determine how the output is to be collated. See Table 2.8 for examples of typical contents. This parameter is ignored if the value of Collate (PLR3, pp. 417–418) is false.</td>
</tr>
<tr>
<td>Fold</td>
<td>integer</td>
<td>A code specifying whether and when to fold the output:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0  Do not fold.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1  Fold at device deactivation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2  Fold at the end of the job.</td>
</tr>
</tbody>
</table>
3 Fold after each page set, as defined by the **Collate** parameter (PLR3, p. 419).

4 Fold after each **showpage** or **copypage** operation.

**FoldDetails** dictionary A dictionary containing device-specific parameters that determine how the output is to be folded. This parameter is ignored if the value of **Fold** is **false**.

**Trim** integer A code specifying whether and when to trim the output:

- 0 Do not trim.
- 1 Trim at device deactivation.
- 2 Trim at the end of the job.
- 3 Trim after each page set, as defined by the **Collate** parameter (PLR3, p. 419).
- 4 Trim after each **showpage** or **copypage** operation.

**Staple** integer A code specifying whether and when to staple the output:

- 0 Do not staple.
- 1 Staple at device deactivation.
- 2 Staple at the end of the job.
- 3 Staple after each page set, as defined by the **Collate** parameter (PLR3, p. 419).
- 4 Staple after each **showpage** or **copypage** operation.

**StapleDetails** dictionary A dictionary containing device-specific parameters that determine how the output is to be stapled. This parameter is ignored if the value of **Staple** is **false**.

**Booklet** boolean A flag specifying whether to fold, trim, and staple the output into booklet form.

**BookletDetails** dictionary A dictionary containing device-specific parameters that determine how the output is to be folded, trimmed, and stapled into booklet form. This parameter is ignored if the value of **Booklet** is **false**.

**Bind** integer A code specifying whether and when to bind the output:

- 0 Do not bind.
- 1 Bind at device deactivation.
- 2 Bind at the end of the job.
3 Bind after each page set, as defined by the **Collate** parameter (PLR3, p. 419).

4 Bind after each **showpage** or **copypage** operation.

**BindDetails** dictionary
A dictionary containing device-specific parameters that determine how the output is to be bound. This parameter is ignored if the value of **Bind** is `false`.

**Punch** integer
A code specifying whether and when to punch the output:

0 Do not punch.

1 Punch at device deactivation.

2 Punch at the end of the job.

3 Punch after each page set, as defined by the **Collate** parameter (PLR3, p. 419).

4 Punch after each **showpage** or **copypage** operation.

**PunchDetails** dictionary
A dictionary containing device-specific parameters that determine how the output is to be punched. This parameter is ignored if the value of **Punch** is `false`.

**Laminate** boolean
A flag specifying whether to laminate the output page. How a page is laminated is device-dependent.

**SlipSheet** integer
A code specifying whether and when to insert a slip sheet into the output:

0 Do not insert slip sheets.

1 Insert a slip sheet at device deactivation.

2 Insert a slip sheet at the end of the job.

3 Insert a slip sheet after each page set, as defined by the **Collate** parameter (PLR3, p. 419).

4 Insert a slip sheet after each **showpage** or **copypage** operation.

A slip sheet may be, for example, a sheet of colored paper whose purpose is to separate multiple copies visually; the selection of specific media is device-dependent. No visible marks can be rendered on a slip sheet; its imageable area is set to an empty region. A slip sheet may pass through the device’s usual imaging mechanism (such as the fuser assembly on a laser printer) or may be routed via a separate paper path. Compare the **InsertSheet** parameter (PLR3, p. 402.)

---

Table 2.8 shows, in addition to the required **Type** entry, another typical **Collate-Details** entry commonly used by PostScript devices. This is only an example; a
given output device may include other entries in its `CollateDetails` dictionary, and may or may not include the one described here. See product-specific documentation for a complete description of the dictionary’s contents for any particular device, as well as those of other details dictionaries (`FoldDetails`, `BookletDetails`, `BindDetails`, `StapleDetails`, `PunchDetails`). Note also that additional details dictionaries not currently supported, such as `TrimDetails` or `LaminateDetails`, could be included if needed by a particular product.

**TABLE 2.8 Typical entries in a CollateDetails dictionary**

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>integer</td>
<td><em>(Required)</em> A code that determines the format and meaning of the other entries in the dictionary.</td>
</tr>
<tr>
<td>ProofSet</td>
<td>boolean</td>
<td>A flag specifying whether to generate a proof copy of the output job. If the value of this parameter is <code>true</code>, the device will print an initial copy of the collated job for the user’s inspection and will then remain in a “hold” state until some device-specific action is taken, such as confirming or aborting the printing of the remaining copies.</td>
</tr>
</tbody>
</table>

**2.2.5 Error Handling**

Table 2.9 describes an additional page device parameter related to error handling, supplementing that described in Section 6.2.7 of the `PostScript Language Reference`, Third Edition.

**TABLE 2.9 Additional page device parameter related to error handling**

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExitJamRecovery</td>
<td>boolean</td>
<td>A flag specifying whether pages that jam in the exit path should be reprinted. Disabling jam recovery may improve performance by allowing a greater degree of overlap in page processing.</td>
</tr>
</tbody>
</table>

**2.3 Envelope Orientation in User Space**

This section describes how default user space is oriented relative to the flap on an envelope. The following discussion assumes that the device’s `Install` procedure (`PLR3`, Section 6.2.6) does not alter the default transformation matrix.
As described in PLR3, p. 401, the page device parameter `PageSize` holds an array of two numbers, \([\text{width} \ \text{height}]\), specifying the overall dimensions of the physical output medium. If the medium is in portrait orientation \((\text{width} < \text{height})\), the default user space is set up as shown in Figure 2.1. The dashed lines in the figure indicate that the flap of the envelope is on the side facing down. In Figure 2.1A, the flap lies along the long edge of the envelope; in Figure 2.1B it is along the short edge. In both cases, the default user space is set up with its origin on the edge of the envelope opposite the flap and in the corner diagonally opposite the return address (on a U. S. business envelope). If the `PageSize` parameter specifies landscape orientation \((\text{width} > \text{height})\), the default user space is rotated 90 degrees counterclockwise from the portrait orientation shown in the figure.
CHAPTER 3

Interpreter Parameters

THIS CHAPTER SUPPLEMENTS Appendix C of the PostScript Language Reference, Third Edition. It provides information about additional user, system, and device parameters.

Much of this chapter concerns communication between an imaging system and a host. These terms are defined as follows:

• An imaging system is a computer system containing a PostScript interpreter that is in direct control of a print engine or other raster output device, and that receives PostScript jobs to execute via one or more communication channels. The communication channels might be, for example, serial or parallel ports, or (for network communications) Ethernet connections.

• A host is a computer system, such as a personal computer or workstation, connected to an imaging system through one or more communication channels. The host—via an application, or perhaps more specifically a driver or spooler—sends PostScript jobs over the communication channel to the imaging system, which executes the jobs.

• The term unit is used in this chapter to refer more broadly to any device connected to a network (an imaging system or some other type of device).
### 3.1 User Parameters

The following user parameters are described in *PLR3*, Table C.1:

<table>
<thead>
<tr>
<th>參數</th>
<th>MaxFontItem</th>
<th>MaxSuperscreen</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccurateScreens</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HalftoneMode</td>
<td>MaxFontItem</td>
<td>MaxUPathItem</td>
</tr>
<tr>
<td>IdiomRecognition</td>
<td>MaxLocalVM</td>
<td>MinFontCompress</td>
</tr>
<tr>
<td>JobName</td>
<td>MaxOpStack</td>
<td>VMReclaim</td>
</tr>
<tr>
<td>MaxDictStack</td>
<td>MaxPatternItem</td>
<td>VMThreshold</td>
</tr>
<tr>
<td>MaxExecStack</td>
<td>MaxScreenItem</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1 describes two additional user parameters: **JobTimeout** and **WaitTimeout**.

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JobTimeout</strong></td>
<td>integer</td>
<td>The <em>job timeout</em>—that is, the number of seconds a job is allowed to execute before it will be aborted and a PostScript <em>timeout</em> error generated. The value of <strong>JobTimeout</strong> is decremented during the job; reading it returns the number of seconds remaining before the job timeout will occur. Time spent waiting for communications and correcting device error conditions is not considered part of the job execution time. The minimum value for <strong>JobTimeout</strong> is 15; supplying a number in the range 1 to 14 will set the parameter to 15. (Allowing smaller values could prevent a subsequent job from successfully setting <strong>JobTimeout</strong> to another value.) A negative value will be ignored and the previous setting of <strong>JobTimeout</strong> used. Setting this parameter to 0 disables job timeout altogether (essentially signifying that the timeout is infinite). Alterations to <strong>JobTimeout</strong> are not subject to <em>save</em> and <em>restore</em>. It is initialized to the value of the <strong>JobTimeout</strong> system parameter at the beginning of each job.</td>
</tr>
<tr>
<td><strong>WaitTimeout</strong></td>
<td>integer</td>
<td>The <em>wait timeout</em>—that is, the number of seconds the interpreter waits to receive additional data from the host before it aborts the current job by executing a PostScript <em>timeout</em> error. A negative value will be ignored and the previous setting of <strong>JobTimeout</strong> used. Setting this parameter to 0 disables wait timeout altogether (essentially signifying that the timeout is infinite).</td>
</tr>
</tbody>
</table>
3.2 System Parameters

The following system parameters are described in PLR3, Table C.2:

<table>
<thead>
<tr>
<th>BuildTime</th>
<th>FontResourceDir</th>
<th>MaxSourceList</th>
</tr>
</thead>
<tbody>
<tr>
<td>ByteOrder</td>
<td>GenericResourceDir</td>
<td>MaxStoredScreenCache</td>
</tr>
<tr>
<td>CurDisplayList</td>
<td>GenericResourcePathSep</td>
<td>MaxUPathCache</td>
</tr>
<tr>
<td>CurFontCache</td>
<td>LicenseID</td>
<td>PageCount</td>
</tr>
<tr>
<td>CurFormCache</td>
<td>MaxDisplayAndSourceList</td>
<td>PrinterName</td>
</tr>
<tr>
<td>CurOutlineCache</td>
<td>MaxDisplayList</td>
<td>RealFormat</td>
</tr>
<tr>
<td>CurPatternCache</td>
<td>MaxFontCache</td>
<td>Revision</td>
</tr>
<tr>
<td>CurScreenStorage</td>
<td>MaxFormCache</td>
<td>StartJobPassword</td>
</tr>
<tr>
<td>CurSourceList</td>
<td>MaxImageBuffer</td>
<td>StartupMode</td>
</tr>
<tr>
<td>CurStoredScreenCache</td>
<td>MaxOutlineCache</td>
<td>SystemParamsPassword</td>
</tr>
<tr>
<td>CurUPathCache</td>
<td>MaxPatternCache</td>
<td></td>
</tr>
<tr>
<td>FactoryDefaults</td>
<td>MaxScreenStorage</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2 describes additional system parameters.

**TABLE 3.2 System parameters**

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompressImageSource</td>
<td>boolean</td>
<td>A flag specifying whether a compression filter will be applied to the image source data where it would benefit memory usage or performance.</td>
</tr>
<tr>
<td>CurBufferType</td>
<td>name</td>
<td><em>(Read-only)</em> A name object specifying how raster memory is used:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Band: The system will render to bands (groups of scan lines), regardless of the amount of RAM available.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hybrid: If the MaxRasterMemory system parameter is large enough to contain a full-page frame buffer, the interpreter will render into the full-page buffer; otherwise, it will render to bands.</td>
</tr>
<tr>
<td>CurInputDevice</td>
<td>string</td>
<td><em>(Read-only)</em> The name of the communications device corresponding to the current input file for the currently executing PostScript job. This value corresponds to the Communication parameter set whose values are normally stored in RAM—for example, %Serial%. See Section 3.3.3, “Communications Parameter Sets.”</td>
</tr>
</tbody>
</table>
**CurOutputDevice**

*string*  
*(Read-only)* Similar to **CurInputDevice**, but corresponds to the current output file.

**DoPrintErrors**

*boolean*

A flag specifying whether the built-in error handler for the product is enabled. All PostScript imaging systems have an error handler to catch errors that are generated by programs (see PLR3, Section 3.11.2). Some imaging systems also have a built-in error handler that can be enabled to print information about the error; **DoPrintErrors** is present only in imaging systems that have such an error handler. Any imaging system that supports a LaserJet 4 or later emulation will have one, which can be controlled not only with this parameter but also with the PJL command

```
@PJL [SET | DEFAULT] LPARM:POSTSCRIPT PRTPSERRS = [ON | OFF]
```

The **Error** dictionary will contain information about the error that an implementation might want to report in its printed error message, such as the contents of the operand, execution, and dictionary stacks at the time of the error. For an example, see Appendix A of *PostScript Language Program Design*.

**DoStartPage**

*boolean*

A flag specifying whether the start page should be printed during system startup. Some printers print a start page when the power is turned on to identify the printer and how it is configured.

**EnvironmentSave**

*boolean*

A flag that applies to systems with multiple page description language (PDL) interpreters and specifies whether permanent objects belonging to all the interpreters persist across PDL switches. If this flag is *false*, the permanent objects do not persist, and all memory belonging to dormant PDL interpreters can potentially be reclaimed when the system runs out of memory. In low-memory configurations, this parameter should always be *false*, since setting it to *true* could make the system essentially unusable. See also **MaxPermanentVM**.

When the value of **EnvironmentSave** is changed, the new value is effective immediately (the system does not have to be restarted).

**FatalErrorAddress**

*integer*

The address at which a fatal system software error occurred, or 0 if no such error occurred. A fatal error causes the PostScript output device to stop execution and, in most products, to restart the PostScript interpreter. Before execution is stopped, the address at which the error occurred is stored in this parameter and is transmitted to the host over the communication channel. Thus, a nonzero value of **FatalErrorAddress** indicates that a fatal error occurred earlier.

On some products, if this value is nonzero during system initialization, the address is printed on the start page or possibly on a separate page.
**InstalledRam** integer *(Read-only)* The number of bytes of installed RAM in the product. *InstalledRam* should not be confused with *RamSize*, which is the amount of RAM available to the page description language.

**JobTimeout** integer The value to which the user parameter *JobTimeout* is initialized at the beginning of each job. See *JobTimeout* in Table 3.1 on page 14.

**MaxHWRenderingBuffer** integer The number of bytes to reserve for use by hardware rendering devices, such as PixelBurst or ColorBurst, to store display lists. The memory is permanently allocated during system initialization. Supplying a value that is outside the valid range (which is product-dependent) sets the parameter to the nearest allowable value. The minimum value for this parameter (typically 8192) meets the requirements of the rendering device, and the maximum value is an amount that will not jeopardize execution of a PostScript job.

**MaxPermanentVM** integer The maximum number of bytes of permanent VM that can be downloaded when *EnvironmentSave* is *true*. This is the amount of VM at *save* level 0 defined by unencapsulated PostScript jobs. Any attempt to download more permanent VM than this maximum will generate a *VMerror*. Supplying a value that is less than the smallest allowable value, or less than the current amount of permanent VM (plus a small threshold value), sets the maximum to the smallest allowable value.

*MaxPermanentVM* is present in the system parameter set only if *EnvironmentSave* is defined.

**MaxRasterMemory** integer The maximum number of bytes occupied by raster memory. Unused raster memory is available for use as virtual memory; thus, this parameter makes it possible to trade off raster memory (which allows larger page sizes and higher resolutions) for virtual memory (which allows more downloaded fonts and more complex pages).

*MaxRasterMemory* is consulted only during system initialization; any changes to the value of this parameter will not take effect until then.

**MinBandBuffers** integer The minimum number of band buffers, each the size of a rendered band (group of scan lines), to be allocated from memory set aside as raster memory. The default value depends on the product and the amount of memory installed. This parameter is typically found on imagesetters, where the default is 2 for configurations with up to 32 MB of memory, or 3 otherwise.

**RamSize** integer *(Read-only)* The number of bytes of installed RAM available to the page description language. In some cases, this value might be less than the total amount of installed RAM in the product (*InstalledRam*). For exam-
SourceListBypass: boolean
A flag that affects the one-operand (dictionary) form of the `image` and `imagemask` operators. If this flag is `true`, `image` and `imagemask` may delay reading data from the source identified by a `DataSource` dictionary entry in certain product-specific cases; otherwise, these operators will immediately read the data from the data source. The data source content will determine what image is painted and whether errors are generated; the existence or value of `SourceListBypass` will have no effect on what is painted, but it can improve performance if set to `true`.

ValidNV: boolean
*(Read-only)* A flag specifying whether nonvolatile memory is being used to store values of persistent parameters. The implementation is product-specific. A common approach is that if nonvolatile memory is corrupt during system initialization, the parameters are initialized with factory default values. Subsequent testing is done to confirm whether nonvolatile memory is still corrupt; if it is, `ValidNV` is set to `false`, and parameter values will not be read from and written to nonvolatile memory thereafter.

In many products, if nonvolatile memory is corrupt it is emulated in RAM. The operating behavior is the same, except that persistent parameter values are lost when the imaging system is powered off or restarted (in which case factory defaults are used).

WaitTimeout: integer
The value to which the user parameter `WaitTimeout` is initialized at the beginning of each job. See `WaitTimeout` in Table 3.1 on page 14.

### 3.3 Device Parameters

This section augments the documentation of device parameters provided in Section C.4 of the *PostScript Language Reference*, Third Edition.

*Note: Not all of the device parameters described in this Supplement will be present in every product.*
3.3.1 Device Parameter Sets

Device parameters are grouped into named parameter sets (dictionaries) that describe the configuration of a physical or logical communication channel, storage device, hardware device, or software entity (such as a language emulator). For example, the parameter set named %Serial% contains the parameters for a serial device. Each parameter set known to the setdevparams and currentdevparams operators corresponds to an instance of the IODevice resource category. Even if two products have devices with the same name, their respective parameter sets might differ because the hardware support for that device may differ on the two products.

Every device parameter set must include a read-only parameter that indicates its type. The key for this parameter is Type, and its value is one of the following four values:

- Communications or Parameters. Adobe has defined various device parameter sets of these two types to aid system administrators in setting up and maintaining network imaging systems. These parameter sets are based on layers within the OSI (Open Systems Interconnection) model, an international standard for how messages should be transmitted between any two points in a telecommunications network, as shown in Figure 3.1. Communications parameter sets are associated with the application layer of OSI or with point-to-point connections that need not be layered or conform to OSI. Parameters parameter sets correspond to the transport, network, data link, and physical layers of OSI, as well as to miscellaneous devices such as the print engine and the calendar.

- FileSystem. Parameter sets of this type correspond to storage devices.

- Emulator. Parameter sets of this type allow PostScript imaging systems to emulate other imaging systems.

Communications, Parameters, and FileSystem parameter sets are described in detail in this chapter. Emulator parameter sets are described in Appendix C.
3.3.2 Device Parameter Interdependencies

Device parameters differ from user and system parameters in that they are interdependent; that is, the acceptability of a value for a given parameter may depend on the value of another parameter. For example, the serial communications device parameter set contains the parameters Interpreter and Protocol: Interpreter determines which page description language is to be used for an incoming job on that channel, and Protocol determines the communications protocol used to send and receive data. Although PostScript is a valid value for Interpreter and Raw is a valid value for Protocol, the serial channel cannot be configured to have both of these values set at the same time; a configurationerror would occur.

Most interdependencies are among parameters in the same device parameter set. However, there is a dependency among all communications devices that requires both the On and Enabled flags to be true in at least one of the communication channels. There might also be cases of interdependencies between certain device parameter sets. For example, if LocalTalk and a serial channel share the same hardware port on an imaging system, both cannot have their On flag set to true at the same time. If this flag is set to true in one of these channels when it is already true in the other channel, it is set to false in the latter channel.
3.3.3 Communications Parameter Sets

Parameter sets of type Communications are associated with the application layer of OSI, from which the PostScript interpreter or emulator receives its jobs, or with point-to-point connections that need not be layered or conform to OSI. (The OSI model is illustrated in Figure 3.1.)

A raster output device can have various physical communication channels and can use many different protocols over these channels. The hardware choices include:

- For point-to-point connections, serial, parallel (unidirectional or bidirectional), and SCSI bus
- For network communications, LocalTalk, Ethernet, and TokenRing

In addition, network adapters, such as line multiplexers, parallel-to-Ethernet adapters, and SCSI-to-Ethernet adapters, allow raster output devices to connect to networks to which they do not have a physical connection.

For point-to-point communications, there are no underlying device parameters specified by Parameters parameter sets (that is, associated with the lower levels of the OSI model); only the Communications parameter set is required. For network communications, however, a combination of Communications and Parameters parameter sets is required to fully specify all layers of the OSI model. The network communications parameter sets apply to raster output devices that are true peers of the network and have a direct connection to it; the Parameters parameter sets support the underlying communications protocols.

To reside directly on networks, raster output devices must recognize various standard communications protocols, including Novell NetWare (SPX/IPX), AppleTalk, and TCP/IP. The choices of protocols and the physical media on which they reside are limited, as shown in Figure 3.2. The connectors in this diagram indicate that the network protocols can be used with the physical medium indicated to deliver and receive messages.
The protocols illustrated in Figure 3.2 are described in further detail below. For more information on the books referred to here, see the Bibliography.

- **Novell SPX/IPX.** The Novell NetWare communications protocols were originally derived from the XNS (Xerox Network System) protocols. For example, the Novell IPX (Internetwork Packet Exchange) protocol is virtually identical to the Xerox Network Layer Protocol called IDP (Internetwork Datagram Protocol). For a discussion of Novell networks, see *Analyzing Novell Networks*.

- **AppleTalk.** AppleTalk is the name Apple® Computer uses for its networking software that is built into every Macintosh® computer. The data link protocol LocalTalk Link Access Protocol (LLAP) is used for communicating over LocalTalk or PhoneNET networks. EtherTalk Link Access Protocol (ELAP) and TokenTalk Link Access Protocol (TLAP) are the data link protocols for communication over Ethernet and TokenRing, respectively. AppleTalk over Ethernet is called EtherTalk, and AppleTalk over TokenRing is called TokenTalk. Datagram Delivery Protocol (DDP), a layer on top of the data link protocol, provides delivery of AppleTalk packets across networks. See *Computer Networks* for a description of TokenRing and Ethernet, and *Inside AppleTalk* for a description of AppleTalk.

- **TCP/IP (or UDP/IP).** TCP/IP is a network architecture that is sponsored by DARPA (the Defense Advanced Research Projects Agency) and has been adopted by a large number of vendors. TCP (Transmission Control Protocol) is a transport layer protocol, and IP (Internet Protocol) is the network layer protocol used by TCP. UDP (User Datagram Protocol) is a connectionless protocol;
it is also dependent on IP for routing to the destination but does not ensure that the destination receives the packets.

**Setting Up Communications Parameter Sets**

The communications parameters for a product can be set up in different ways and may involve the use of front panels, hardware switches, and PostScript operators and procedures. This section describes a generic model for setting communications parameters that works across a variety of products and enables PostScript spoolers and utilities to use the same model when reading and writing communications device parameters.

A raster output device typically has several hardware ports for communications. There can be multiple physical ports of the same type, in which case each has its own parameter set. When there are multiple instances of a particular Communications parameter set, the naming convention for instances beyond the first is to add the letter B for the second instance, C for the third, and so on. If there were two serial ports, for example, their parameter sets would be named %Serial% and %SerialB%.

Furthermore, multiple parameter sets can be associated with the same physical port if that port can serve different roles, such as the serial port named channel B in an imaging system that has the following three ports:

- A parallel port, associated with the parameter set named %Parallel%.
- A serial port named channel A, wired to a 25-pin RS-232A connector and associated with the parameter set named %Serial%.
- A port named channel B, wired to either an 8-pin or a 9-pin connector and associated with two parameter sets named %SerialB% and %LocalTalk%. This port can serve as either a serial port or a LocalTalk port, depending on which of the two parameter sets has its On parameter set to true.

**%CommName%, %CommName_NV%, and %CommName_Pending%**

To provide a consistent model that is product-independent, all communications devices have three parameter sets. If the name of the device is, for example, %Serial%, the three associated parameter sets are named %Serial%, %Serial_NV%, and %Serial_Pending%. These names are shown in general here as
%%%CommName%, %CommName_NV%, and %CommName_Pending% (where CommName is the part of the name that varies by device). These three parameter sets have the following general characteristics:

- %CommName% values are usually stored in RAM and do not persist when the imaging system is powered off.
- %CommName_NV% values are usually stored in nonvolatile memory. If a product has limited nonvolatile memory, only some of the %CommName_NV% values may actually be saved to nonvolatile memory; otherwise, typically all writeable %CommName_NV% values are saved to nonvolatile memory. If the intent is to affect persistent values, PostScript utility programs should use %CommName_NV%, and the implementation will do the best it can given the amount of nonvolatile memory available in the product.
- %CommName_Pending% is a read-only parameter set whose values are used to configure the communications hardware and software at the beginning of the next file received from a communication channel. This parameter set reflects either the current values of a writeable parameter set, such as %CommName%, or some predetermined values selected via a switch or front panel. How the system computes the values in %CommName_Pending% is described in the next section.

In our imaging system example above, the following parameter sets might be available:

- %Parallel%
- %Parallel_NV%
- %Parallel_Pending%
- %Serial%
- %Serial_NV%
- %Serial_Pending%
- %SerialB%
- %SerialB_NV%
- %SerialB_Pending%
- %LocalTalk%
- %LocalTalk_NV%
- %LocalTalk_Pending%

Note: On some products, the parameter sets %CommName%, %CommName_NV%, and %CommName_Pending% may not be distinct from one other.

Hierarchy of Communications Parameter Sets

Figure 3.3 illustrates the hierarchical relationship that exists among the three parameter sets %CommName%, %CommName_NV%, and %CommName_Pending%.
As shown in this diagram, values written to `%CommName%` are written through to `%CommName_Pending%`, and values written to `%CommName_NV%` are written through to `%CommName%` and then to `%CommName_Pending%`. Beyond this, there are several variables:

- The product may have a front panel. The values set by the user at the front panel are written to `%CommName%` or to `%CommName_NV%` (if the values are to persist across restarts and power cycles). Some products store to only one of these sets.

- The product may have switches through which it can be directed to use either `%CommName_NV%` parameter sets or built-in (hard-wired) values. Generally, products do not have both a front panel and switches.

- PostScript programs (usually spoolers or utilities) may write parameter values to `%CommName%` or `%CommName_NV%` (usually the former) at any time. This is true whether the output device has a front panel or switches.

The `%CommName%` parameter set, which is in RAM and does not persist when the imaging system is powered off, is used in many cases (but not all) to update the `%CommName_Pending%` set. Thus, on many products, such as those with a front panel but no switches, the `%CommName%` and `%CommName_Pending%` sets always have the same values and appear redundant.

In general, a spooler or utility almost always should write to `%CommName%`. It should write to `%CommName_NV%` only if parameters are to persist when the imaging system is powered off.
A front panel usually writes to /%CommName_NV%/ to change the power-on parameters, although it can also write to /%CommName%/.

**Multiple Nonvolatile Parameter Sets (/%CommName_NVn%/)**

It is possible to have multiple nonvolatile parameter sets, named /%CommName_NV%/, /%CommName_NV2%/, /%CommName_NV3%/, and so on. As with a single nonvolatile set, these parameter sets obtain their values by being written to by a PostScript spooler or utility.

Figure 3.4 shows a situation in which there are three nonvolatile parameter sets. Only one of these sets can be active at any given time. In this figure, the active set is /%CommName_NV2%/, as indicated by the switch setting. When the switch is set to this position, or when the product is restarted or powered on with the switch in this position, the values in /%CommName_NV2%/ are written through to /%CommName%/ and to /%CommName_Pending%/.

While the setting /%CommName_NV2%/ is active, a PostScript job can write to any of the nonvolatile parameter sets, but only values written to /%CommName_NV2%/ would migrate to /%CommName%/ and /%CommName_Pending%/.

Changing the switch to the position corresponding to /%CommName_NV3%/ would cause /%CommName_NV3%/ values to become the active values in /%CommName%/ and /%CommName_Pending%/.

**Predetermined (Hard-Wired) Parameter Values**

In addition to the switch settings that indicate which nonvolatile parameter set should be used, other switch settings can affect the hierarchy of parameter sets and cause a predetermined set of communications parameters to be written directly to /%CommName_Pending%/.

In Figure 3.5, for example, switch positions 1 and 2 designate two “hard-wired” parameter sets. When the switch is set to position 1, for example, PostScript programs may still write to one of the /%CommName_NV%/ parameter sets or to /%CommName%/,

but not to /%CommName_Pending%/ unless the switch is reset to one of positions 3 through 5.
FIGURE 3.4 Communications parameter sets using NV values

FIGURE 3.5 Communications parameter sets using “hard-wired” values
This example explains the existence of the \%CommName\_Pending\% set as separate from the \%CommName\% set: that is, it allows absolute determination of the communications parameters that will be used, no matter what other activity occurs.

Note that reading the \%CommName\_NV\% or \%CommName\% set gives no information about the parameters being used for the current job or the next job, but simply returns the values last written to these sets. Reading \%CommName\_Pending\% returns the values to be used for the next job. Determining the parameters of the current job is of little interest: either the job is a page description, in which case it should not be accessing device parameters at all, or it is a utility that is interested in either determining or affecting the settings for future jobs. If the device parameters are used as described above, utilities can be written without concern for exactly which parameters are stored in nonvolatile memory or for whether a utility job, front panel, or switch is used to establish communications parameters.

As described in the previous section, a spooler or utility almost always should write to \%CommName\%. It should write to \%CommName\_NV\% only if parameters are to persist across restarts and power cycles.

Changes to Communications parameters sets take effect after the current file (containing one or more PostScript jobs) is fully processed by the interpreter and before the next file is read.

Once a file that specifies communications changes has been interpreted, a transition to new settings of the \%CommName\_Pending\% set occurs. No additional data should be transferred from the host to the imaging system on this communication channel until after the transition to new settings is complete.

**Note:** There is a distinction between “end-of-job” and “end-of-file,” since a file can contain multiple jobs; see PLR3, Section 3.7.7.

**Entries Common to All Communications Parameter Sets**

Table 3.3 describes the entries found in all device parameter sets of type Communications.
### TABLE 3.3 Device parameters common to all Communications parameter sets

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be Communications.</td>
</tr>
<tr>
<td>HasNames</td>
<td>boolean</td>
<td>(Read-only) A flag specifying whether the device represented by the parameter set (that is, the communication channel) supports named files. This parameter is always false in Communications parameter sets.</td>
</tr>
<tr>
<td>Enabled</td>
<td>boolean</td>
<td>A flag specifying whether data arriving on the communication channel should be considered as a job to be scheduled for execution by the PostScript interpreter or emulator. If this flag is true, arriving data will be scheduled as an executable job; otherwise, the data will not be scheduled as an executable job, but the channel can be used directly by a job for reading and writing data. A configurationerror occurs in either of the following situations:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Attempting to set Enabled to true when On is false in the same parameter set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Attempting to set Enabled to false in one parameter set when it would result in all channels having Enabled set to false</td>
</tr>
<tr>
<td>On</td>
<td>boolean</td>
<td>A flag specifying whether the communication channel is turned on and able to send and receive data. If this flag is true, data transmitted to the channel by a host is buffered and flow control protocols are applied; otherwise, data sent to the channel is lost. A configurationerror occurs if an attempt is made to set On to false in either of the following situations:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• When Enabled is true in the same parameter set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• When it would result in all channels having On set to false</td>
</tr>
</tbody>
</table>

If two communications devices share the same physical port and On is set to true when it is already true for another channel associated with this port, the one that was originally On is turned off and disabled, and the new one is turned On.

If On is true and Enabled is false, the channel is not considered a source of jobs to be scheduled, but the channel can be used by a PostScript job to send and receive data by means of the file operators.

When the system is powered on, if it is determined that all installed communication channels are currently off, it is up to the product to perform its own unique recovery strategy. For example, the product could search for an installed communication channel and force it on, even if this was not the state preserved in nonvolatile memory. An alternative would be to inform the user...
via the operator control panel that the product cannot be initialized until the problem is rectified.

**DelayedOutputClose**  boolean

A flag specifying how the output channel is managed after each job finishes executing. This parameter is set independently for each communication channel. When this flag is true:

- An EOF (end-of-file) indication is not sent until all pages of a job have been printed. On network channels, the connection remains open until the job finishes printing.
- If a job produces output and if previous jobs using the same output channel have not finished printing, the output will not be sent until those jobs have completed printing and the EOF for each of them has been sent.
- Spontaneous messages, such as imaging system error messages, are sent to the channel whether it is the output channel for the currently executing job or the output channel for previous jobs that have not finished printing.

When the value of **DelayedOutputClose** is false:

- An EOF is sent as soon as a job finishes executing in the interpreter. On network channels, the connection may be closed when the job finishes executing, even though pages produced by the job might not have finished printing.
- Output generated by a job can be transmitted without delay, even if previous jobs using the same output channel have not finished printing. (The EOF for each of those jobs will have already been sent.)
- Spontaneous messages are sent to the channel only if it is the output channel for the job currently executing, and even if it is the output channel for previous jobs that have not finished printing.

The **DelayedOutputClose** setting for a job source is controlled by the parameter sets for the output channel of that job source. Thus, if the serial B channel is used for the output of jobs received on the parallel port, the **DelayedOutputClose** value in the %SerialB% parameter set applies to jobs received on both the serial B port and the parallel port.

The **DelayedOutputClose** parameter does not appear in a Communications parameter set if the channel has no output or if all messages generated asynchronously from the interpreter are directed to a logically separate channel.

**Note:** In versions prior to 2014, upon completion of each job the interpreter waits for all pages of the job to be printed before sending an EOF or closing a connection and before starting to execute another job. Thus, any output associated with the job, including imaging system error messages, is always sent before the next job begins and before the EOF is sent or a connection closed.
**Interpreter** name

A name object specifying which interpreter or emulator is to be used to interpret the next incoming job arriving on the communication channel. This parameter is used only if **Enabled** is true and **PrinterControl** is PSPrinter. For certain communication channels there is a relationship between the Interpreter and Protocol parameters that can result in a configurationerror; see Protocol in Table 3.4.

Interpreter (or Protocol) can be set without a password if there are no other entries in the dictionary passed to setdevparams. If the only additional entry in the dictionary is the password, it is ignored.

The valid values of Interpreter are PostScript, AutoSelect, Diablo630, EpsonFX850, HP7475A, LaserJetIII, LaserJetIIIP, PCL, and ProprinterXL. The values Diablo630, EpsonFX850, HP7475A, LaserJetIII, LaserJetIIIP, and ProprinterXL refer to imaging systems (no longer sold) that offered a page description language (PDL) other than PostScript. The value PCL refers to Hewlett-Packard’s Printer Control Language; this value was first introduced when the LaserJet 4 printer was emulated, and it continues to reflect the imaging systems of Hewlett-Packard. The imaging system being emulated is described in an instance of the PDL resource category. PCL was first introduced as a value of Interpreter when the LaserJet 4 printer was emulated, and it continues to reflect the imaging systems of Hewlett-Packard. For more information on emulation, see Appendix C.

The Interpreter value of AutoSelect provides automatic and seamless switching between the available interpreters and emulators based on the input data stream. Interpreter should be set to AutoSelect on channels that connect to hosts that alternately send PostScript jobs, raw PCL, and “printscreen” jobs (in the IBM PC–compatible environment). In general, it can be used on any communication channel, but it requires that the underlying communications protocol be one that preserves all incoming data. For serial and parallel communication channels:

- If Protocol is set to Raw (the recommended setting when the value of Interpreter is AutoSelect), AutoSelect detects interpreter, job, and protocol boundaries and automatically selects the protocol. When it detects that a PostScript job is being received, only Normal and TBCP protocols can be recognized (Binary is not supported).

- If Protocol is set to TBCP or Binary, AutoSelect detects interpreter boundaries and job boundaries.

- Attempting to set Interpreter to AutoSelect when Protocol is Normal (which reserves certain control characters for communications functions) causes a configurationerror.

For other communication channels that are binary in nature:
• AutoSelect detects interpreter boundaries and job boundaries.

• The Interpreter value of PCL is used only in imaging systems that emulate a LaserJet 4 or later LaserJet printers. For emulations of earlier LaserJet models, the values LaserJetIP and LaserJetIII are used.

**PrinterControl**

A name object that indicates how a host queries and controls the imaging system when accessing it via this communication channel. The valid values of PrinterControl are PSpriinter and PJL. If PrinterControl is set to PSpriinter:

• The Interpreter parameter selects the page description language (PDL) interpreter.

• Imaging system error messages are sent in the usual Adobe fashion (on channels processing jobs, and in %% [...] %% format).

• PJL (Printer Job Language) commands are recognized only if the value of Interpreter is AutoSelect or PCL. If Interpreter is AutoSelect, the ENTER LANGUAGE and UEL commands are handled and other PJL commands are identified as such and discarded. If Interpreter is PCL, only the UEL command is handled.

If PrinterControl is set to PJL:

• The selection of the PDL interpreter is controlled by PJL, either by explicit invocation of the PJL ENTER LANGUAGE command or based on the value of the PJL variable PERSONALITY. The value of PERSONALITY can be PCL, POSTSCRIPT, or AUTO, where AUTO provides automatic, seamless switching between interpreters.

• Imaging system error messages are in PJL format and are enabled or disabled by PJL commands. Whether a job is being processed on a channel does not affect whether messages are sent.

• The “PJL current environment” is used on each invocation of a PDL to set up the initial state.

If PrinterControl is set to PSpriinter, a Communications parameter set specifies the interpreter or emulator, error messages conform to PostScript guidelines, and an initial graphics state is established at the start of each job. If PrinterControl is set to PJL, the behavior emulates LaserJet communications, which rely on PJL to specify the PDL interpreter needed, the structure of error messages, and the initial graphics state.

PJL is described in the Hewlett-Packard Printer Job Language Reference Manual; see the Bibliography for details.
Device Parameters

Point-to-Point Communications Parameter Sets

In addition to the entries common to all Communications parameter sets (Table 3.3 above), serial, parallel, and SCSI device parameter sets include the entries listed in the following tables. For more information on these protocols, see Adobe Technical Notes #5009, *Adobe Serial and Parallel Communications Protocols Specification*, and #5010, *Adobe SCSI Input Protocol Specification*.

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>name</td>
<td>A name object specifying the communications protocol that is used:</td>
</tr>
<tr>
<td>Normal</td>
<td></td>
<td>Certain control characters are reserved for communications functions, such as end-of-file, software flow control, abort job, and status query, and so cannot be carried as data. This protocol is suitable only for ASCII-encoded PostScript jobs, and not for PostScript jobs containing binary data or for any printer emulation jobs.</td>
</tr>
<tr>
<td>Raw</td>
<td></td>
<td>All characters are treated as data; there are no reserved characters, and no communications functions are available. Normally, this protocol is suitable for use only with an emulator and not with the PostScript interpreter. However, in products that support an <strong>Interpreter</strong> value of <strong>AutoSelect</strong>, protocol processing is handled by the <strong>AutoSelect</strong> facility; therefore, the value of <strong>Protocol</strong> should be <strong>Raw</strong> when <strong>Interpreter</strong> is set to <strong>AutoSelect</strong>.</td>
</tr>
<tr>
<td>TBCP</td>
<td></td>
<td>Tagged Binary Communication Protocol. An encoding scheme allows the full range of 8-bit values to be transmitted as data, while also providing for certain communications functions. In addition, it provides explicit begin-protocol and end-protocol sequences that permit the receiver to switch automatically between <strong>Normal</strong> and <strong>TBCP</strong> processing. This protocol is suitable for use with any language (the PostScript interpreter or an emulator).</td>
</tr>
<tr>
<td>Binary</td>
<td></td>
<td>An encoding scheme allows the full range of 8-bit values to be transmitted as data, while also providing for certain communications functions. This protocol is suitable for use with any language; however, it is obsolete and has been superseded by <strong>TBCP</strong>.</td>
</tr>
</tbody>
</table>

A **configurationerror** occurs if setting either the **Protocol** or the **PrinterControl** parameter would result in the combination of a **Protocol** value
of Normal and a PrinterControl value of PJL. A configuration error also occurs if setting Protocol or Interpreter would result in one of the following combinations when Enabled is true and PrinterControl is PSPrinter:

- A Protocol value of Raw and an Interpreter value of PostScript
- A Protocol value of Normal and an Interpreter value other than PostScript

In other words, PostScript jobs cannot be executed over a channel using the Raw protocol, and emulators cannot be executed over a channel using the Normal protocol. Likewise, when automatic selection of interpreters and emulators is chosen, the Normal protocol cannot be used.

Protocol (or Interpreter) can be set without a password if there are no other entries in the dictionary passed to setdevparams. If the only additional entry in the dictionary is the password, it is ignored.

FlowControl name A name object specifying the serial flow control method used between the host and the device:

- **Dtr**
  - DTR (data terminal ready) and DSR (data set ready) hardware signals are used by the imaging device and the host, respectively, to indicate to each other when data may be transmitted. A high value for the signal indicates that data may be transmitted; a low value indicates that data should not be transmitted.

- **DtrLow**
  - Same as Dtr except the active sense of the signals is reversed: a low signal indicates that data may be transmitted, a high signal indicates that data should not be transmitted.

- **EtxAck**
  - Two control characters, ETX (end-of-text) and ACK (acknowledgment), are reserved for flow control usage. The protocol is symmetrical for the imaging device and the host. Each sender knows an agreed-upon maximum number of bytes that the other side can receive, and the sender may send up to this number of bytes followed by an ETX. The sender may send more data only after receiving an ACK from the other side.

- **XonXoff**
  - Used for communicating with PCs (non-UNIX systems). Two control characters, XON and XOFF, are reserved for flow control usage.

  For all Protocol values except Raw, the protocol is symmetrical for the imaging device and the host. If one side wants the other to stop sending data, it sends an XOFF;
when it is ready to receive data again, it sends an XON. When the On parameter is set to false, the interpreter sends an XOFF to the host just before turning off the channel.

If the value of Protocol is Raw, XON and XOFF sent from the host to the imaging system are treated as data and not reserved as flow control characters. XON and XOFF sent from the imaging system to the host are treated as flow control characters.

XonXoff2 Used for communicating with UNIX systems; similar to XonXoff except that when the On parameter is set to false, the interpreter sends an XON to the host. This allows the host to unload any data it wants to send; the interpreter simply drops the data.

RobustXonXoff Similar to XonXoff except that periodically (typically every second) the interpreter will send the host an XON if it is able to receive data.

Note: Not all serial channels support all flow control methods.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud</td>
<td>integer</td>
<td>The baud rate on the underlying serial hardware. Normally this parameter can be set to any nonnegative value; however, the underlying serial hardware will round it to the nearest achievable baud rate. Hardware rounding will not be reflected in the value of the parameter when it is read. On some products this parameter might be restricted to a small number of valid values.</td>
</tr>
<tr>
<td>DataBits</td>
<td>integer</td>
<td>The number of data bits per byte communicated over the channel. Valid values are 7 and 8. If the value is 7, the high bit of a received byte of data is set to 0.</td>
</tr>
<tr>
<td>StopBits</td>
<td>integer</td>
<td>The number of stop bits that are transmitted by the serial hardware. Valid values are 1 and 2; the hardware will always be able to receive data transmitted with one or two stop bits.</td>
</tr>
<tr>
<td>Parity</td>
<td>name</td>
<td>A name object specifying the parity to be used between the host and the device when CheckParity is true:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>None Neither the host nor the device should send a parity bit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Space The parity bit should always be 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mark The parity bit should always be 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Even Even parity should be used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Odd Odd parity should be used.</td>
</tr>
</tbody>
</table>
The total number of bits for each byte transmitted or received is the sum of the number of start bits (always 1), data bits (7 or 8), parity bits (0 or 1), and stop bits (1 or 2). Most serial devices do not support 8-bit data with either Space or Mark parity; setting the parameters in this way does not generate a configuration error, but the results of such a configuration are unpredictable.

CheckParity boolean A flag specifying whether the device should do parity checking on incoming data. If CheckParity is true and a parity error occurs, an ioerror results. This parameter is ignored if Parity specifies None.

---

**TABLE 3.5 Additional device parameters specific to the %Parallel% (or %ParallelX%) parameter set**

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>name</td>
<td>See Table 3.4 above.</td>
</tr>
</tbody>
</table>
| Handshake   | integer| A code identifying the hardware/software signal interface that is to be used for communications across the parallel interface:

0  Unidirectional communications, commonly used by PCs and PC-compatibles. This is the default.
1  Bidirectional communications, as specified by version 0.6 of the Hewlett-Packard Boise specification. This is an obsolete value; it was superseded by 2, which was later superseded by 6 through 8.
2  IEEE-1284 Draft Specification 1.00. This is an obsolete value, superseded by 6, 7, and 8.
3  Unidirectional, ACK in Busy.
4  Unidirectional, ACK after Busy (Japanese implementation).
5  Unidirectional, ACK while Busy.
6  IEEE-1284 Draft Specification 2.00, ACK in Busy.
7  IEEE-1284 Draft Specification 2.00, ACK after Busy.
8  IEEE-1284 Draft Specification 2.00, ACK while Busy.

*Note:* The IEEE-1284 Draft Specification 2.00 is the IEEE standard 1284-1994; see the Bibliography for more information.

Attempting to set Handshake to one of the unidirectional values (0, 3, 4, or 5) when OutputDevice designates a parallel communications device generates a configuration error.

OutputDevice string The name of the communications device to be used for %stdout and %stderr—that is, %Serial%, %SerialB%, %SerialC%, and so on, or %Parallel%,
Device Parameters

%ParallelB%, %ParallelC%, and so on. If **OutputDevice** is the empty string, %stdout and %stderr information is routed to a product-specific default output device.

Attempting to set **OutputDevice** to a parallel communications device when the value of **Handshake** is one of the unidirectional values (0, 3, 4, or 5) generates a **configuration error**.

**nAckPulseWidth**  integer  The width (duration), in nanoseconds, of the **nAck** pulse, rounded to the nearest value that can be achieved. **nAck** is a signal that originates from the imaging system to the host in the form of a pulse; its meaning depends on which **Handshake** mode is being used or what state the imaging system’s device driver is currently in. The **nAck** pulse width is controlled by the imaging system’s support circuitry or the device driver and is normally in the range 500–10,000.

**nStrobeExpectedPulseWidth**  integer  The expected duration, in nanoseconds, of the **nStrobe** pulse, rounded to the nearest value that can be achieved. **nStrobe** is a signal that originates from the host to the imaging system in the form of a pulse; its meaning depends on which **Handshake** mode is being used or what state the imaging system’s device driver is currently in. The **nStrobe** pulse width, which may vary from host to host, is normally in the range 750–500,000.

---

**TABLE 3.6**  Additional device parameters specific to the %ScsiComm% (or %ScsiCommX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bus</strong></td>
<td>string</td>
<td>(Read-only) The name of the SCSI bus device parameter set associated with this channel. Valid values are %Scsi%, %ScsiB%, %ScsiC%, and so on.</td>
</tr>
<tr>
<td><strong>Filtering</strong></td>
<td>name</td>
<td>A name object indicating whether the input stream needs further filtering before the data can be correctly interpreted as a page description language:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>None Passes the data unchanged to the interpreter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>InterpreterBased Filters the input stream as necessary to conform to the language. For example, the data stream may have been sent to the imaging system encoded as a TBCP PostScript job and must be decoded to a normal PostScript job before being passed to the interpreter. See Protocol in Table 3.4 on page 33 for a description of TBCP.</td>
</tr>
</tbody>
</table>

**Note:** In a complete AppleTalk/Macintosh environment, the **Filtering** parameter should be set to None, or communications problems will result.
Network Communications Parameter Sets

Network communications rely on a combination of Communications and Parameters parameter sets. The Communications parameter sets correspond to the application layer of the OSI model and connect directly to a PostScript interpreter or emulator, which receives its jobs from and sends messages back to the host using the devices associated with these parameter sets.

The network communications parameter sets, described in more detail below, are: %LPR%, %AppSocket%, and %Telnet%, which allow for the use of the TCP/IP protocol over Ethernet; %RemotePrinter% and %PrintServer%, which are associated with the Novell NetWare application layer; and %LAT%, used primarily for communication with terminal servers. This section also describes the device parameter sets for the AppleTalk networking software (LocalTalk, EtherTalk, and TokenTalk), which do not split out the physical layer.

- %LPR%. The UNIX command lpr sends a printer job to a printer system. The lpr service depends on the TCP/IP protocol; TCP port 515 is used for it. Because lpr (or the lpr daemon) is by definition unidirectional, any %stdout or %stderr information is transmitted by means of the Syslog facility (described in Table 3.17 on page 48).

- %AppSocket%. The AppSocket protocol was created to support TranScript® software from Adobe Systems. It provides a more robust interface than %LPR% because it uses bidirectional communications directly. AppSocket can also be used by drivers other than TranScript and for transmitting data other than PostScript jobs. For more information about AppSocket, contact Adobe Developer Support.

- %Telnet%. Telnet, which is a TCP/IP network service, gives network users interactive access to and exclusive use of the PostScript interpreter or emulator. TCP port 23 is used for Telnet.

- %RemotePrinter%. The Novell remote printer application is managed by a Novell print server and takes print jobs downloaded from a print server. A remote imaging system may be shared by many print servers.

- %PrintServer%. The Novell print server application communicates with Novell file servers to download print jobs from the print queues. A print server may communicate with multiple file servers and access multiple print queues.

- %LAT%. The LAT (Local Area Transport) protocol is used primarily for communication between hosts and terminal servers. Often, these terminal servers
have a printing device attached, making it possible to print from a given set of hosts to a set of imaging systems on a local area network.

- **%LocalTalk%**. LocalTalk is the name given to AppleTalk running over LocalTalk or PhoneNET networks.

- **%EtherTalk%**. EtherTalk is the name given to AppleTalk running over Ethernet networks.

- **%TokenTalk%**. TokenTalk is the name given to AppleTalk running over Token-Ring networks.

In addition to the entries common to all Communications parameter sets (Table 3.3 on page 29), the above device parameter sets include the entries listed in the following tables, with one exception: there are no additional entries in the **%Telnet%** (or **%TelnetX%**) parameter set.

**TABLE 3.7 Additional device parameters specific to the %LPR% (or %LPRX%) parameter set**

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering</td>
<td>name</td>
<td>See Table 3.6 on page 37.</td>
</tr>
</tbody>
</table>
| PrintHost        | string| A string of up to 63 characters specifying at most two IP mask/address pairs. The mask is applied to the IP address to specify which hosts are allowed to make lpr connections. The format of this string is:

\[
[mask1]\text{[/]}address1 [[mask2]\text{[/]}address2]
\]

where everything in square brackets is optional. The format of an IP mask or address is \text{N.N.N.N}, where each N is a decimal number in the range 0–255. If an address is specified without a corresponding mask, a mask of 255.255.255.255 is assumed.

For example,

138.46.25.37

is a mask/address pair in which the mask is not specified, whereas

255.255.255.0/138.46.25.37 255.255.255.0/138.46.24.38

is two mask/address pairs, each specifying a mask.

Specifying an invalid IP address, such as 0.0.0.0, 127.0.0.0, or N.N.N.255, will result in a rangecheck error.

| SendWindowSize  | integer | The send window size (the number of bytes that the imaging system will transmit in a data packet). This parameter provides a means of tuning the |
code for optimal throughput. Its setting is applied at imaging system startup, when memory is allocated for use by the network communications software. The actual window size is established when the connection is opened and may be smaller than this parameter’s value in order to accommodate the host’s expectations.

The window size specified here overrides any request for this parameter in the associated sets of type Parameters—for example, %TCP%. The valid range of values for this parameter is 1024–65535.

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReceiveWindowSize</td>
<td>integer</td>
<td>The receive window size (the number of bytes that the host will transmit in a data packet). See SendWindowSize for additional information.</td>
</tr>
</tbody>
</table>
StatusPortNumber can refer to the same port as ControlPortNumber, but a configuration error will occur if it refers to the same port as DataPortNumber.

DataPortNumber integer

The port number of a bidirectional port for transmission of printer jobs. The suggested default value is 9100. A different port number may be used to avoid a conflict if another unit on the network is already using 9100. If DataPortNumber refers to the same port as either ControlPortNumber or StatusPortNumber, a configuration error will occur.

---

**TABLE 3.9** Additional device parameter specific to the %RemotePrinter% (or %RemotePrinterX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering</td>
<td>name</td>
<td>See Table 3.6 on page 37.</td>
</tr>
</tbody>
</table>

**TABLE 3.10** Additional device parameters specific to the %PrintServer% (or %PrintServerX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering</td>
<td>name</td>
<td>See Table 3.6 on page 37.</td>
</tr>
<tr>
<td>LoginPassword</td>
<td>string</td>
<td>A string of up to 32 characters specifying the password that the print server application uses to gain access to the job queue. Setting this parameter to the empty string indicates that no password has been specified. The value of LoginPassword when queried is always the string INVALID; attempts to set LoginPassword to this string will be ignored.</td>
</tr>
<tr>
<td>PreferredServer</td>
<td>string</td>
<td>The name of the file server that the print server attempts to attach to in order to service print queues. The validity of the PreferredServer name is not checked; the value is passed directly to the nearest file server for routing request if the file server specified by PreferredServer does not respond. Novell 3.x file server names are limited to 47 characters; spaces and the characters /* +,|;:=&lt; &gt;?{] are not allowed. Novell 4.x file server names can contain up to 64 characters; spaces are allowed but not desirable. Novell NetWare 4.x converts spaces to underscores. (Strings that contain spaces may not work on Novell NetWare 3.x.)</td>
</tr>
</tbody>
</table>
TABLE 3.11 Additional device parameters specific to the %LAT% (or %LATX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering</td>
<td>name</td>
<td>See Table 3.6 on page 37.</td>
</tr>
<tr>
<td>Physical</td>
<td>string</td>
<td><em>(Read-only)</em> The physical layer over which LAT is accessed; a device parameter set corresponding to a physical communications medium, such as %EthernetPhysical%. This parameter can associate one network layer parameter set with one and only one physical layer parameter set.</td>
</tr>
<tr>
<td>Groups</td>
<td>string</td>
<td>The groups allowed to access this printer device. A group is specified by an integer in the range 0–255. A range of groups is designated with a hyphen, and multiple groups or ranges are separated by spaces. For example, the string 3-8 145-160 200 designates two ranges and one additional group. The default value is the empty string, which gives all groups access.</td>
</tr>
<tr>
<td>KeepaliveTimer</td>
<td>integer</td>
<td>The interval (in seconds from 0 to 180) at which the circuit layer exchanges keep-alive messages to maintain circuits when no session traffic is present.</td>
</tr>
<tr>
<td>MulticastTimer</td>
<td>integer</td>
<td>The interval (in seconds from 0 to 180) at which directory service advertisements (SAs) are multicast to advertise the printer’s service.</td>
</tr>
<tr>
<td>RetransmitTimer</td>
<td>integer</td>
<td>The interval (in seconds from 0 to 10) at which the circuit layer retransmits unacknowledged messages.</td>
</tr>
<tr>
<td>RetransmitLimit</td>
<td>integer</td>
<td>The number of retransmissions (from 4 to 120) that will be attempted before a circuit is declared dead.</td>
</tr>
</tbody>
</table>

TABLE 3.12 Additional device parameters specific to the %LocalTalk% (or %LocalTalkX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering</td>
<td>name</td>
<td>See Table 3.6 on page 37.</td>
</tr>
<tr>
<td>NodeID</td>
<td>integer</td>
<td><em>(Read-only)</em> The local network address of the device. Valid addresses are in the range 128–254. A value of 0 indicates that the address has not been established. The value of this parameter is used as an address hint when addresses are first established as part of the AppleTalk protocol. As such, the parameter might not represent the actual address until that portion of the protocol is complete during initialization of the AppleTalk device.</td>
</tr>
<tr>
<td>LocalTalkType</td>
<td>string</td>
<td>The <em>type</em> component of the AppleTalk entity name. The entity name consists of the three components <em>zone</em>, <em>type</em>, and <em>object</em>, each a string of up to 32 characters. The <em>zone</em> component is set to the wildcard character (*), and the <em>object</em> component is set to the value of the <em>PrinterName</em> system parameter.</td>
</tr>
</tbody>
</table>
The conventional value of \texttt{LocalTalkType} for any PostScript-capable printer is LaserWriter. Imaging systems having this type are automatically available for direct use by any host. A spooler or other system administration program can change the value of \texttt{LocalTalkType} to make the imaging system inaccessible for direct use.

The \texttt{appletalktype} compatibility value will reflect a change to the \texttt{LocalTalkType} parameter. If the imaging system also supports EtherTalk or TokenTalk communications, setting \texttt{LocalTalkType} will set the \texttt{EtherTalkType} or \texttt{TokenTalkType} parameter to the same value.

### TABLE 3.13 Additional device parameters specific to the \texttt{%EtherTalk\%} (or \texttt{%EtherTalk\ X\%}) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering</td>
<td>name</td>
<td>See Table 3.6 on page 37.</td>
</tr>
<tr>
<td>NodeID</td>
<td>integer</td>
<td>(Read-only) The local network address of the device. Valid addresses are in the range 1–254. A value of 0 indicates that the address has not been established. The value of this parameter is used as an address hint when addresses are first established as part of the AppleTalk protocol. As such, the parameter might not represent the actual address until that portion of the protocol is complete during initialization of the AppleTalk device.</td>
</tr>
<tr>
<td>EthernetAddress</td>
<td>string</td>
<td>(Read-only) The Ethernet address of the imaging system. The format is \texttt{xx:xx:xx:xx:xx:xx}, where each \texttt{x} is a hexadecimal digit (0–9 or either A–F or a–f).</td>
</tr>
<tr>
<td>EtherTalkType</td>
<td>string</td>
<td>The \texttt{type} component of the EtherTalk entity name. The entity name consists of the three components \texttt{zone}, \texttt{type}, and \texttt{object}, each a string of up to 32 characters. The \texttt{zone} component is set to the value of the \texttt{EtherTalkZone} parameter, and the \texttt{object} component is set to the value of the \texttt{PrinterName} system parameter. The \texttt{appletalktype} compatibility value will reflect a change to the \texttt{EtherTalkType} parameter. If the imaging system also supports LocalTalk or TokenTalk communications, setting the \texttt{EtherTalkType} string will set the \texttt{LocalTalkType} or \texttt{TokenTalkType} parameter to the same value.</td>
</tr>
<tr>
<td>EtherTalkZone</td>
<td>string</td>
<td>The \texttt{zone} portion of the EtherTalk entity name. See \texttt{EtherTalkType}.</td>
</tr>
</tbody>
</table>

### TABLE 3.14 Additional device parameters specific to the \texttt{%TokenTalk\%} (or \texttt{%TokenTalk\ X\%}) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering</td>
<td>name</td>
<td>See Table 3.6 on page 37.</td>
</tr>
</tbody>
</table>
NodeID integer (Read-only) See Table 3.13 above.

Address string (Read-only) The TokenRing address of the imaging system. The format is \texttt{xxxxxxxxxxxxx}, where each \texttt{x} is a hexadecimal digit (0–9 or either A–F or a–f).

Bridging name A name object specifying how to do bridging on the TokenRing network:

- **Adaptive**: The software will automatically recognize the routing style and respond in kind (either as a one-time determination or each time it processes a connection). This is the default value.
- **Transparent**: The entire “universe” is one large single ring structure and all identities are unique.
- **SourceRoute**: Routing is done by specifying an explicit path, including the ring identification, or Routing Information Field (RIF).

TokenTalkType string The \texttt{type} component of the TokenTalk entity name. The entity name consists of the three components \texttt{zone}, \texttt{type}, and \texttt{object}, each a string of up to 32 characters. The \texttt{zone} component is set to the value of the \texttt{Zone} parameter, and the \texttt{object} component is set to the value of the \texttt{PrinterName} system parameter.

The \texttt{appletalktype} compatibility value will reflect a change to the \texttt{TokenTalkType} parameter. If the imaging system also supports LocalTalk or EtherTalk communications, setting the \texttt{TokenTalkType} parameter will set the \texttt{LocalTalkType} or \texttt{EtherTalkType} parameter to the same value.

Zone string The \texttt{zone} component of the TokenTalk entity name. See \texttt{TokenTalkType}.

3.3.4 Parameters Parameter Sets

Parameter sets of type Parameters are used to configure the following:

- The SNMP and Syslog network services
- The lower layers of network communications
- The SCSI bus and the IDE bus
- The print engine, the operator console, and the calendar

For network communications, these parameter sets correspond to the transport, network, data link, and physical layers of the OSI model, as shown in Figure 3.1 on page 20. (The PostScript interpreter or emulator receives its jobs from the ap-
3.3 Device Parameters

For each parameter set associated with an implementation of the network layer, there must be a parameter set associated with the data link and physical layers (in other words, a distinct interface to the network). For example, if there is only one network layer implementation but it is connected to $n$ network interfaces, then for each network layer parameter set there is a unique data link and physical parameter set named within it. A parameter set for the network layer can be viewed as a distinct binding of the network address and the network interface.

The Parameters parameter sets are described in more detail below. Except where indicated otherwise in these descriptions, changes to parameters in the parameter set do not take effect until the imaging system is reinitialized. Note that as with Communications parameter sets, when there are multiple instances of a particular Parameters parameter set, the naming convention for instances beyond the first is to add the letter B for the second instance, C for the third, and so on—for example, %Scsi% and %ScsiB% if more than one SCSI bus is present.

- **%SNMP%**: SNMP (Simple Network Management Protocol) allows the system administrator to query for information about the unit. The information that can be queried is driven by a database called a Management Information Base (MIB). The parameters listed for the %SNMP% parameter set in this Supplement are only those that need to be accessible from the PostScript language as opposed to the MIB (or MIBs) the device may have. These are the only parameters that are changeable from an environment separate from SNMP (the network side). The rules governing when changes to parameters in this parameter set take effect are included in the description of each parameter. More information about SNMP is available in Internet Engineering Task Force Request for Comments (IETF RFC) 1157.

- **%Syslog%**: Syslog is a logging facility that sends log messages back to a UNIX host. Most of the messages contain network-specific information, but they may include any other pertinent information the unit wants to convey. Communication is via the UDP transport layer.

- **%TCP%**: TCP is the transport layer responsible for reliable data transfer; it is a connection-oriented IP-based protocol that guarantees message delivery and reception. Any packet that is lost will be retransmitted.

- **%UDP%**: UDP is the connectionless IP-based protocol. When UDP is used with a peer host, there is no need for handshaking prior to communication. UDP
packets are sent without any guarantee of delivery and may arrive at the destination in any order.

• %IP%. IP is the network layer responsible for routing messages to their destinations. This layer decides which physical interface is to send outgoing messages and which transport layer is to receive incoming messages.

• %SPX%. SPX is the Novell NetWare connection-oriented protocol. When SPX is used to communicate with a peer host, handshaking takes place before the connection is ready. The delivery of SPX packets is expected to be acknowledged to guarantee delivery, and packets arrive in sequence at the destination. Unlike TCP, it does not provide a sliding window functionality (that is, the ability to temporarily change the data packet size) for flow control.

• %IPX%. IPX is the Novell NetWare connectionless (or datagram) protocol. When IPX is used with a peer host, there is no need for handshaking prior to communication. IPX packets are sent without any guarantee of delivery and may arrive at the destination in any order. NetWare broadcasting uses IPX.

• %EthernetPhysical% and %TokenRingPhysical%. These correspond to a physical Ethernet or TokenRing connector, respectively, along with its associated hardware and the data link layer software that handles events from this device.

• %Scsi% and %Ide%. These are used to configure the SCSI bus or IDE bus, respectively. The parameter set is always present in an imaging system that has that type of bus, even if no devices are present on the bus.

• %Engine%. This parameter set is used to configure the print engine. The rules governing when changes to parameters in this parameter set take effect are product-dependent, but typically the effect is immediate.

• %Console%. This parameter set provides a means of setting and controlling characteristics of the operator console of an output device that includes the PostScript language. The parameters currently defined in this set provide an extensible way of selecting the natural language (for example, English or Japanese) in which information will be displayed on the operator console. The rules governing when changes to parameters in this parameter set take effect are product-dependent, but typically the effect is immediate.

• %Calendar%. A printer has a battery-powered time-of-day clock. This clock must be set initially, and then twice a year to follow daylight saving time (if applicable). The string %Calendar% identifies the calendar device, and the entries in the dictionary describe the local date and time. Changes to parameters in this parameter set take effect immediately.
Required entries for the parameter sets of type Parameters are described in the tables that follow in this section. Some of these parameter sets include the specification of a node address, a unique address for a node on a network. The form of the node address depends on the protocol being used to communicate with the node. Table 3.15 lists the various forms a node address can take.

### TABLE 3.15 Forms of a node address

<table>
<thead>
<tr>
<th>PROTOCOL</th>
<th>ADDRESS FORM</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Novell IPX/SPX</td>
<td>XXXXXXXXXXXxxxxx</td>
<td>Each X or x is a hexadecimal digit (0–9 and either A–F or a–f). The 8-digit XXXXXXXX is the network part of the address, and the 10-digit xxxxxxxxxx is the MAC (media access control) part, known as the Novell node number.</td>
</tr>
<tr>
<td>AppleTalk DDP</td>
<td>NNn</td>
<td>Each N or n is a decimal number in the range 0–255. NN is the network part of the address, and n is the node ID.</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>NNNNN</td>
<td>An IP address. Each N is a decimal number in the range 0–255.</td>
</tr>
</tbody>
</table>

### TABLE 3.16 Device parameters in the %SNMP% (or %SNMP%X) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>SysName</td>
<td>string</td>
<td>A string of up to 32 characters specifying the name of the imaging system (expected by SNMP). Changes to this parameter take effect immediately.</td>
</tr>
<tr>
<td>SysLocation</td>
<td>string</td>
<td>A string of up to 32 characters specifying the location of the imaging system. Changes to this parameter take effect immediately.</td>
</tr>
<tr>
<td>SysContact</td>
<td>string</td>
<td>A string of up to 32 characters that traditionally specifies the name and either the phone number or the address of the person responsible for the imaging system. Changes to this parameter take effect immediately.</td>
</tr>
<tr>
<td>PrivateHost</td>
<td>string</td>
<td>A string of up to 49 characters specifying one node address per protocol—the address of a host that is able to set those SNMP variables that can be set. The various forms of a node address are listed in Table 3.15 above. If there is more than one such address, they are separated by spaces. The empty string, indicating that no host has access, is the usual default value; the imaging system will therefore need to have this parameter set explicitly with the setdevparams operator before SNMP can be used. Changes to this parameter do not take effect until the imaging system is reinitialized.</td>
</tr>
</tbody>
</table>
**CHAPTER 3**

**Table 3.17 Device Parameters in the %Syslog% (or %SyslogX%) Parameter Set**

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>LogHost</td>
<td>string</td>
<td>The IP address for a host that receives Syslog messages from the unit. For the format of an IP address, see Table 3.15 on page 47. The empty string indicates that no Syslog messages are to be sent by the unit. Specifying an IP address equal to 0.0.0.0 or N.N.N.255 will result in a rangecheck error.</td>
</tr>
<tr>
<td>LogPriority</td>
<td>integer</td>
<td>The lowest priority of logging messages that are to be sent on to the Syslog host. All messages of this priority or higher (where smaller values designate higher priority) are sent. The priorities, listed below from highest to lowest, follow the BSD UNIX and SunOS™ convention.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>
6 Informational messages
7 Debugging messages

### TABLE 3.18 Device parameters in the %TCP% (or %TCPX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>On</td>
<td>boolean</td>
<td>A flag specifying whether the TCP protocol is activated at imaging system startup.</td>
</tr>
<tr>
<td>SendWindowSize</td>
<td>integer</td>
<td>See Table 3.7 on page 39.</td>
</tr>
<tr>
<td>ReceiveWindowSize</td>
<td>integer</td>
<td>See Table 3.7 on page 39.</td>
</tr>
</tbody>
</table>

### TABLE 3.19 Device parameters in the %UDP% (or %UDPX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>On</td>
<td>boolean</td>
<td>A flag specifying whether the UDP protocol is activated at imaging system startup.</td>
</tr>
<tr>
<td>Checksum</td>
<td>boolean</td>
<td>A flag specifying whether checksum values will be inserted in outgoing packets formed by the software. The default value should be true.</td>
</tr>
</tbody>
</table>

### TABLE 3.20 Device parameters in the %IP% (or %IPX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>On</td>
<td>boolean</td>
<td>A flag specifying whether the IP protocol is activated at imaging system startup.</td>
</tr>
<tr>
<td>Physical</td>
<td>string</td>
<td>(Read-only) The name of the physical layer over which the protocol is accessed. The string designates a device parameter set corresponding to a physical communications medium, such as %EthernetPhysical%. This parameter can associate a network layer parameter set with one and only one physical layer parameter set.</td>
</tr>
<tr>
<td>TransmitEncapsulation</td>
<td>name</td>
<td>A name object specifying the transmit encapsulation type:</td>
</tr>
</tbody>
</table>
SNAP 802.2 or 802.5 header with a SNAP header. This is the default value when Physical is %TokenRingPhysical%.

DIX (Ethernet only) Ethernet II header. This should be the default value when Physical is %EthernetPhysical%.

**Note:** These values were introduced to eliminate dependencies on the type of connection used (token-ring or Ethernet). The valid values in version 2016 were 802.3-2-SNAP, 802.5-2-SNAP, and DIX.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPAddressDynamic</td>
<td>boolean A flag controlling how the IPAddress and related parameters (BroadcastAddress, NetworkMask, and GatewayAddress) are assigned. If IPAddressDynamic is true, these parameters are obtained automatically from the network during imaging system startup, using the BOOTP or RARP protocol. If IPAddressDynamic is false, these parameters are set explicitly using setdevparams. A change to the value of IPAddressDynamic takes effect at the next imaging system initialization. An invocation of setdevparams that changes IPAddress or related parameters must also set IPAddressDynamic to false.</td>
</tr>
<tr>
<td>IPAddress</td>
<td>string The IP address of the imaging system. For the format of an IP address, see Table 3.15 on page 47. This address is mapped to the imaging system's physical layer address as specified by the Physical parameter (for example, EthernetAddress if Physical is %EthernetPhysical%). The default value is the empty string, which means that the IP protocol layer is not active. The value of IPAddress can be set explicitly at any time; however, it will take effect only at the next imaging system initialization, and only if the IPAddressDynamic parameter is set to false in the same setdevparams call that sets the value of IPAddress. Specifying an IP address equal to 127.0.0.0, 127.0.0.255, or any address whose first field is in the range 224–255 will result in a rangecheck error.</td>
</tr>
<tr>
<td>NetworkMask</td>
<td>string A mask indicating which fields of the value of the IPAddress parameter designate the network portion of the IP address and which designate the node portion. The format is the same as that of an IP address, as described in Table 3.15 on page 47. For example, 255.255.255.0 is the NetworkMask value for a class B network with subnets. NetworkMask is used to determine whether a certain IP address is on the same network as the imaging system. The value of NetworkMask can be set explicitly at any time; however, it will take effect only at the next imaging system initialization, and only if the IPAddressDynamic parameter is set to false in the same setdevparams call that sets the value of NetworkMask.</td>
</tr>
</tbody>
</table>
If the value of NetworkMask is not valid with respect to the value of IPAddress, it will be changed to a value that is valid, with no warning to the user. For example, if a class A network mask is given with a class B IP address, NetworkMask will be changed to the default class B network mask. The default masks are 255.0.0.0, 255.255.0.0, or 255.255.255.0 for class A, B, and C, respectively; no subnets are accounted for in these masks.

**BroadcastAddress**

The broadcast address mask used when broadcasting messages to the local network. The format is the same as that of an IP address, as described in Table 3.15 on page 47.

The value of BroadcastAddress can be set explicitly at any time; however, it will take effect only at the next imaging system initialization, and only if the IPAddressDynamic parameter is set to false in the same setdevparams call that sets the value of BroadcastAddress.

If the value of BroadcastAddress is not valid with respect to the values of IPAddress and NetworkMask, it will be changed to a value that is valid, with no warning to the user. For example, suppose IPAddress is 134.14.15.16 and BroadcastAddress is 134.14.255.255; if the user changes IPAddress to 134.15.15.16 without explicitly changing BroadcastAddress, BroadcastAddress will be changed to 134.15.255.255.

**GatewayAddress**

The destinationAddress/gatewayAddress pairs to other networks, where the addresses are IP addresses separated by a slash and multiple address pairs are separated by spaces. The number of address pairs depends on the product. For the format of an IP address, see Table 3.15 on page 47. Loopback addresses (127.N.N.N) and broadcast addresses (N.N.N.255) will result in a rangecheck error.

destinationAddress is the network address. Class A network addresses require the first field to be nonzero; the other fields may be zero (for a network with no subnets) or may contain subnet address information. Class B or class C network addresses require the first two or three fields, respectively, to be nonzero.

A network address of 0.0.0.0 is a special value that is used if none of the specified network addresses matches the desired target IP address. If the same destination address is specified in more than one address pair, all but the last such pair will be ignored.

The empty string specifies dynamic routing, which means that a Routing Information Protocol (RIP) request to the network is used to gather routing information. If dynamic routing is available in the product, this should be the default value.
The value of **GatewayAddress** can be set at any time; however, it will take effect only at the next imaging system initialization, and only if the **IPAddressDynamic** parameter is set to `false` in the same `setdevparams` call that sets the value of **GatewayAddress**. If **IPAddressDynamic** is `true`, **GatewayAddress** should be set to appropriate valid values (and **IPAddressDynamic** set to `false`, as required), since dynamic routing is not always reliable when the **IPAddress** value is received via RARP.

**TABLE 3.21** Device parameters in the %SPX% (or %SPXX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td><em>(Read-only)</em> The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>On</td>
<td>boolean</td>
<td>A flag specifying whether the SPX protocol is activated at imaging system startup.</td>
</tr>
<tr>
<td>ReceiveWindowSize</td>
<td>integer</td>
<td>See Table 3.7 on page 39. The valid range of values for this parameter is 1024–59392.</td>
</tr>
</tbody>
</table>

**TABLE 3.22** Device parameters in the %IPX% (or %IPXX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td><em>(Read-only)</em> The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>On</td>
<td>boolean</td>
<td>A flag specifying whether the IPX protocol is activated at imaging system startup.</td>
</tr>
<tr>
<td>Checksum</td>
<td>boolean</td>
<td>A flag specifying whether checksum values will be inserted in outgoing packets formed by the software. The default value should be <code>true</code>.</td>
</tr>
<tr>
<td>Physical</td>
<td>string</td>
<td><em>(Read-only)</em> See Table 3.20 on page 49.</td>
</tr>
<tr>
<td>TransmitEncapsulation</td>
<td>name</td>
<td>A name object specifying the transmit encapsulation type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NO_SNAP</strong> 802.2 or 802.5 header without a SNAP header. This should be the default value when <strong>Physical</strong> is <code>%TokenRingPhysical%</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>SNAP</strong> 802.2 or 802.5 header with a SNAP header.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>802.3</strong> <em>(Ethernet only)</em> 802.3 header. This should be the default value when <strong>Physical</strong> is <code>%EthernetPhysical%</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>DIX</strong> <em>(Ethernet only)</em> Ethernet II header.</td>
</tr>
</tbody>
</table>
**Adaptive** Indicates that, by the nature of the interaction between host and imaging system, the encapsulation format to use in responses to the host can be derived at imaging system startup.

The value of this parameter is checked solely for validity; it is not checked for applicability.

*Note:* These values were introduced to eliminate dependencies on the type of connection used. The valid values in version 2016 were 802.3-2, 802.3-X, 802.3-2-SNAP, 802.5-2-SNAP.

The values of **TransmitEncapsulation** and **Physical** for the various Novell frame types are as follows:

<table>
<thead>
<tr>
<th>Novell frame type</th>
<th>TransmitEncapsulation</th>
<th>Physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet_802.3</td>
<td>802.3</td>
<td>%EthernetPhysical%</td>
</tr>
<tr>
<td>Ethernet_802.2</td>
<td>NO_SNAP</td>
<td>%EthernetPhysical%</td>
</tr>
<tr>
<td>TOKEN_RING</td>
<td>NO_SNAP</td>
<td>%TokenRingPhysical%</td>
</tr>
<tr>
<td>Ethernet_SNAP</td>
<td>SNAP</td>
<td>%EthernetPhysical%</td>
</tr>
<tr>
<td>TOKEN_RING_SNAP</td>
<td>SNAP</td>
<td>%TokenRingPhysical%</td>
</tr>
<tr>
<td>Ethernet_II</td>
<td>DIX</td>
<td>%EthernetPhysical%</td>
</tr>
<tr>
<td>Any</td>
<td>Adaptive</td>
<td>%EthernetPhysical%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or %TokenRingPhysical%</td>
</tr>
</tbody>
</table>

**NetworkAddress** string *(Read-only)* The address of the network in which the unit is located. The format is XXXXXXXX, where each X is a hexadecimal digit. The concatenation of the **NetworkAddress** value and the Novell node number forms the node address that will uniquely identify the unit on the network. The **NetworkAddress** value is obtained from the Novell file server on the local network at imaging system startup. The Novell node number is derived from the MAC (media access control) address of the networking media. For the Ethernet, the Novell node number is the value of the **EthernetAddress** parameter of the %EthernetPhysical% set.

**HopCount** integer The maximum number of routers the print server will go through in trying to attach to file servers while looking for print queues to service. The preferred value is the smallest value needed to reach all of the imaging system's file servers. A count of 15 is defined as trying to reach all reachable servers. Specifying a negative value or a value larger than 15 will set the count to 15. If the **PreferredServer** parameter is set in the %PrintServer% parameter set, the **HopCount** parameter is ignored unless the server specified by **PreferredServer** is unreachable.
### TABLE 3.23 Device parameters in the %EthernetPhysical% (or %EthernetPhysicalX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>On</td>
<td>boolean</td>
<td>A flag specifying whether the Ethernet channel is enabled at imaging system startup.</td>
</tr>
</tbody>
</table>
| EthernetAddress | string | (Read-only) The Ethernet address of the unit. The format is \(xx:xx:xx:xx:xx:xx\), where each \(x\) is a hexadecimal digit.  

**Note:** When Novell NetWare is used, the Ethernet address (minus its colons) is also the Novell node number.

| ConnectorType | name | (Read-only) A name object specifying which Ethernet connector type is being used. Valid values are RJ45, BNC, AUI, and AAUI. |
| Name          | string | (Read-only) A string of up to 16 characters specifying a mnemonic name for the interface being used—for example, le0 for “lance chip interface unit 0” or so0 for “sonic chip interface unit 0.” |

### TABLE 3.24 Device parameters in the %TokenRingPhysical% (or %TokenRingPhysicalX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>On</td>
<td>boolean</td>
<td>A flag specifying whether the TokenRing channel is enabled at imaging system startup.</td>
</tr>
<tr>
<td>Address</td>
<td>string</td>
<td>(Read-only) The TokenRing address of the unit. The format is (xx:xx:xx:xx:xx:xx), where each (x) is a hexadecimal digit.</td>
</tr>
</tbody>
</table>

| Bridging | name | See Table 3.14 on page 43. |
| ConnectorType | name | (Read-only) A name object specifying which TokenRing connector type is being used. Valid values are RJ45, DB9, and MAU. |
| Name     | string | (Read-only) See Table 3.23 above. |
| Speed    | integer | The speed at which the ring is operated, in megabits per second. Valid values are 4 and 16. |
### TABLE 3.25 Device parameters in the %Scsi% (or %ScsiX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>CheckParity</td>
<td>boolean</td>
<td>A flag specifying whether parity on the SCSI bus is to be checked. The default value is usually true.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> Setting CheckParity to true on products that do not support parity checking is inadvisable.</td>
</tr>
<tr>
<td>BootDelay</td>
<td>integer</td>
<td>(Only when disks are present on the bus) The number of seconds the disk I/O driver should wait during imaging system startup for the disk attached to the imaging system to spin up, before determining that a disk is not present or not responding. A value of 0 means that there is no waiting for the disk to spin up. This parameter should be set in accordance with the characteristics of the disk.</td>
</tr>
<tr>
<td>InitiatorId</td>
<td>integer</td>
<td>(Only when disks are present on the bus) The SCSI bus address used by the imaging system when it serves as initiator. The valid range of values is 0–7. The default value is usually 7.</td>
</tr>
<tr>
<td>TargetId</td>
<td>integer</td>
<td>(Only when a %ScsiComm% communication channel is present on the bus) The SCSI bus address reserved by the imaging system for use as the %ScsiComm% communication channel. This address may be the same as the InitiatorId address. The valid range of values is 0–7.</td>
</tr>
<tr>
<td>Poll</td>
<td>integer</td>
<td>(Only when disks are present on the bus) A bit mask in the range 0–254 indicating which addresses on the SCSI bus should be polled by the imaging system when it looks for disks during system initialization. Bits are numbered from right to left, beginning with 0 for the low-order bit. Bits set to 1 mean that polling should occur; bits set to 0 mean “do not poll.” For example, a 1 in bit 0 means poll for %disk0%. Any bits in the mask that correspond to addresses used as the following should be set to 0: the imaging system’s InitiatorId or TargetId address; the InitiatorId address of other hosts on the bus; or the TargetId address of devices belonging to other hosts on the bus. The value of Poll cannot be 0, since at least the bit corresponding to the InitiatorId address should be set to 1. The default value is usually 127 (16#7F). If a bit is set to poll an address that should not be polled, anomalies may occur on the bus. Specifically, if the bit is set to poll the address corresponding to the imaging system’s InitiatorId or TargetId address, a configuration error will occur.</td>
</tr>
</tbody>
</table>


### TABLE 3.26  Device parameters in the %ide% (or %ideX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>BootDelay</td>
<td>integer</td>
<td>(Only when disks are present on the bus) See Table 3.25 above.</td>
</tr>
<tr>
<td>Poll</td>
<td>integer</td>
<td>(Only when disks are present on the bus) A bit mask in the range 0–3 indicating which addresses on the IDE bus should be polled by the imaging system when it looks for disks during system initialization. For example, a 1 in bit 0 means poll for %disk0%.</td>
</tr>
</tbody>
</table>

### TABLE 3.27  Device parameters in the %Engine% (or %EngineX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>BSizeStandard</td>
<td>name</td>
<td>A name object that determines the physical dimensions of the paper when B4 or B5 paper is selected. (Note that “default unit” in the following descriptions means 1/72 inch.)</td>
</tr>
</tbody>
</table>

**ISO**  The International Organization for Standardization, also known as ISO, defines the “metric” paper sizes (A3, A4, B3, B4, B5, and so on), which are used in Europe and much of the rest of the world. The dimensions for the B4 and B5 paper sizes as defined by ISO are as follows:

- B4 250 × 353 mm or 709 × 1001 default units
- B5 176 × 250 mm or 499 × 709 default units

**JIS**  The Japanese Industrial Standards Committee specifies standards for use in Japan. Japan uses the standard “A” paper sizes but observes a slightly different definition of the “B” paper sizes. The dimensions for the B4 and B5 paper sizes as defined by JIS are as follows:

- B4 257 × 364 mm or 729 × 1032 default units
- B5 182 × 257 mm or 516 × 729 default units

| Darkness | number   | The overall lightness or darkness of the rendered page on a monochrome device. This parameter does not affect the frame buffer, nor does it have any computational overhead. Valid values range from 0.0 to 1.0, where 0.0 means minimum darkness and 1.0 means maximum darkness. This parameter is |
provided in some products whose marking hardware allows software control of colorant application.

**DarknessBlack** number The overall lightness or darkness of the black color on a rendered page produced on a device with multiple toner stations. See Darkness for further information.

**DarknessCyan** number The overall lightness or darkness of the cyan color on a rendered page. See Darkness for further information.

**DarknessMagenta** number The overall lightness or darkness of the magenta color on a rendered page. See Darkness for further information.

**DarknessYellow** number The overall lightness or darkness of the yellow color on a rendered page. See Darkness for further information.

**PageCount** integer The number of pages fed by the print engine. This count includes all the pages that were printed successfully as well as those that were jammed or otherwise spoiled. The value of this parameter is determined by querying the engine.

**TimeToStandby** integer The number of minutes after which the print engine will go into a standby mode, in which it stops trying to keep itself ready to print a page; that is, it stops keeping its fuser hot. The next time the controller sends a feed or pre-feed command, the engine will enter the “warming up” state until it is ready to print.

The range of allowable values for this parameter is typically 0–720 (where 0 never lets the engine go into standby mode). If an invalid value is specified, it will be rounded to the nearest allowable value.

---

### TABLE 3.28 Device parameters in the %Console% (or %ConsoleX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td><em>(Read-only)</em> The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>CharSet</td>
<td>name</td>
<td>A name object specifying either a standard character set or a vendor-specific character set. If a standard set, the name will designate the standard or, if applicable, a variant of the standard (such as ISO-646-IRV, a variant of ASCII). The valid standard character set names are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASCII Basic ASCII (7-bit); currency symbol is $</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ISO-646-IRV Same as ASCII except that an international currency symbol replaces $</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ISO-8859-1 The ISO 8-bit Latin1 character set</td>
</tr>
</tbody>
</table>
The CID-keyed Japanese character collection

If the name is vendor-specific, it should designate the vendor and the identification of the character set used by that vendor—for example, IBM-Codepage-550.

**Language name**

A name object specifying the natural language in which information will be displayed, with a 2-character language code from the ISO 639 standard. Table 3.29 lists a selection of these codes; the complete list may be obtained from the International Organization for Standardization (see the Bibliography).

**Country name**

A name object that indirectly specifies the dialect of the language by specifying the country in which the dialect is used. The country is indicated by a 2-character country code from the ISO 3166 standard. Table 3.30 lists a selection of these codes; the complete list may be obtained from the International Organization for Standardization (see the Bibliography).

### TABLE 3.29 Selected ISO 639 language codes

<table>
<thead>
<tr>
<th>CODE</th>
<th>LANGUAGE</th>
<th>CODE</th>
<th>LANGUAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>Czech</td>
<td>JA</td>
<td>Japanese</td>
</tr>
<tr>
<td>DA</td>
<td>Danish</td>
<td>KO</td>
<td>Korean</td>
</tr>
<tr>
<td>DE</td>
<td>German</td>
<td>NL</td>
<td>Dutch</td>
</tr>
<tr>
<td>EL</td>
<td>Greek</td>
<td>NO</td>
<td>Norwegian</td>
</tr>
<tr>
<td>EN</td>
<td>English</td>
<td>PL</td>
<td>Polish</td>
</tr>
<tr>
<td>FI</td>
<td>Finnish</td>
<td>PT</td>
<td>Portuguese</td>
</tr>
<tr>
<td>FR</td>
<td>French</td>
<td>RU</td>
<td>Russian</td>
</tr>
<tr>
<td>GA</td>
<td>Irish</td>
<td>SK</td>
<td>Slovak</td>
</tr>
<tr>
<td>HU</td>
<td>Hungarian</td>
<td>SV</td>
<td>Swedish</td>
</tr>
<tr>
<td>IT</td>
<td>Italian</td>
<td>ZH</td>
<td>Chinese</td>
</tr>
<tr>
<td>IW</td>
<td>Hebrew</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CODE</td>
<td>COUNTRY</td>
<td>CODE</td>
<td>COUNTRY</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>AR</td>
<td>Argentina</td>
<td>IN</td>
<td>India</td>
</tr>
<tr>
<td>AU</td>
<td>Australia</td>
<td>IT</td>
<td>Italy</td>
</tr>
<tr>
<td>BE</td>
<td>Belgium</td>
<td>JP</td>
<td>Japan</td>
</tr>
<tr>
<td>BO</td>
<td>Bolivia</td>
<td>LU</td>
<td>Luxembourg</td>
</tr>
<tr>
<td>BR</td>
<td>Brazil</td>
<td>MX</td>
<td>Mexico</td>
</tr>
<tr>
<td>CA</td>
<td>Canada</td>
<td>NL</td>
<td>Netherlands (Holland)</td>
</tr>
<tr>
<td>CL</td>
<td>Chile</td>
<td>NO</td>
<td>Norway</td>
</tr>
<tr>
<td>CN</td>
<td>China</td>
<td>NZ</td>
<td>New Zealand</td>
</tr>
<tr>
<td>CO</td>
<td>Colombia</td>
<td>PA</td>
<td>Panama</td>
</tr>
<tr>
<td>DE</td>
<td>Germany</td>
<td>PE</td>
<td>Peru</td>
</tr>
<tr>
<td>DK</td>
<td>Denmark</td>
<td>PH</td>
<td>Philippines</td>
</tr>
<tr>
<td>EC</td>
<td>Ecuador</td>
<td>PK</td>
<td>Pakistan</td>
</tr>
<tr>
<td>ES</td>
<td>Spain</td>
<td>PL</td>
<td>Poland</td>
</tr>
<tr>
<td>FI</td>
<td>Finland</td>
<td>PT</td>
<td>Portugal</td>
</tr>
<tr>
<td>FR</td>
<td>France</td>
<td>PY</td>
<td>Paraguay</td>
</tr>
<tr>
<td>GB</td>
<td>United Kingdom</td>
<td>SA</td>
<td>Saudi Arabia</td>
</tr>
<tr>
<td>GR</td>
<td>Greece</td>
<td>SE</td>
<td>Sweden</td>
</tr>
<tr>
<td>HU</td>
<td>Hungary</td>
<td>TW</td>
<td>Taiwan</td>
</tr>
<tr>
<td>ID</td>
<td>Indonesia</td>
<td>US</td>
<td>United States</td>
</tr>
<tr>
<td>IE</td>
<td>Ireland</td>
<td>VE</td>
<td>Venezuela</td>
</tr>
<tr>
<td>IL</td>
<td>Israel</td>
<td>ZA</td>
<td>South Africa</td>
</tr>
</tbody>
</table>
TABLE 3.31  Device parameters in the %Calendar% (or %CalendarX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be Parameters.</td>
</tr>
<tr>
<td>Running</td>
<td>boolean</td>
<td>A flag specifying whether the clock should be running. Setting the date and time requires turning on the clock (by setting this parameter to true) and simultaneously setting the date and time parameters listed below. When queried, this parameter specifies whether the clock is currently running. If the clock is not running (for whatever reason) or is assumed to be inaccurate (as discovered by internal checks of the clock operation), the Year parameter returns 0 and the other date and time parameters return the minimum values in their respective ranges.</td>
</tr>
<tr>
<td>Year</td>
<td>integer</td>
<td>The year, in the range 1980–2079, or 0 if queried when the clock is not running or is assumed to be inaccurate.</td>
</tr>
<tr>
<td>Month</td>
<td>integer</td>
<td>The month, in the range 1–12.</td>
</tr>
<tr>
<td>Day</td>
<td>integer</td>
<td>The day of the month, in the range 1–31.</td>
</tr>
<tr>
<td>Hour</td>
<td>integer</td>
<td>The hour, in the range 0–23.</td>
</tr>
<tr>
<td>Minute</td>
<td>integer</td>
<td>The minute, in the range 0–59.</td>
</tr>
<tr>
<td>Second</td>
<td>integer</td>
<td>The second, in the range 0–59.</td>
</tr>
</tbody>
</table>

3.3.5 FileSystem Parameter Sets

The four commonly used parameter sets of type FileSystem are:

- %diskn% (where n starts from 0 for the first instance)
- %cartridge% (or %cartridge%n%, where n starts from 1 for the second instance)
- %rom% (or %romn%), denoting a cartridge device that is nonremovable and nonwriteable—that is, some form of ROM
- %ram% (or %ramn%), denoting a file system that is writeable and stored in some form of RAM

The memory used by %ram% competes with other uses of memory, such as PostScript virtual memory, and explicit use of %ram% to store files competes with other uses of %ram%, such as storing data for ReusableStreamDecode filters.
Note: %ram% is required on all version 3010 products. As of version 3011, the access attributes r+, w+, and a+ have been implemented for %ram%. For a description of these attributes, see PLR3, pp. 79–80.

Workstation-based imaging systems that use the native file system have a separate device, os, that provides direct access to that file system; a FileSystem parameter set, %os%, is associated with this device. Most of the parameters in this set are read-only; a minimal set of parameters is provided primarily for consistency with other types of file systems.

As shown in the list above, digit suffixes are used to distinguish between multiple instances of parameter sets for storage devices, and only a disk device parameter set uses the suffix 0. Letter suffixes are generally used only to distinguish between multiple instances of other kinds of FileSystem parameter sets, such as %os%.

Required entries in FileSystem parameter sets are listed in the tables below. Note that in these tables, read-only refers only to access by language operators (for example, setdevparams and currentdevparams). The value of a read-only parameter can change, but not as a result of invoking setdevparams. Changes to parameters in FileSystem parameter sets take effect immediately.

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be FileSystem.</td>
</tr>
<tr>
<td>Bus</td>
<td>string</td>
<td>(Read-only) The name of the bus on which this disk resides—%Scsi% (or %ScsiX%) if a SCSI bus, %Ide% (or %IdeX%) if an IDE bus. If the imaging system uses a file system of the native operating system rather than the Adobe storage device implementation, this parameter may not be meaningful or may be absent. More information about the bus (for this disk device) can be obtained by using currentdevparams with the value of this parameter as the device of interest.</td>
</tr>
<tr>
<td>HasNames</td>
<td>boolean</td>
<td>(Read-only) A flag specifying whether the device supports named files. Since a device will not mount successfully unless it contains a valid file system, HasNames is always true for mounted devices; if false, the device is not mounted.</td>
</tr>
<tr>
<td>Mounted</td>
<td>boolean</td>
<td>A flag specifying whether the device should be mounted or dismounted, or (if queried) whether the device is currently mounted.</td>
</tr>
</tbody>
</table>
Setting `Mounted` to `true` indicates that the system should attempt to mount the device, and setting it to `false` that it should attempt to dismount the device. Mounting a device makes it known to the system and makes it at least readable, depending on the nature of the device. A device will not mount successfully unless it contains a valid file system (that is, unless the `HasNames` parameter is `true`). If an attempted mount (or dismount) fails, a `configurationerror` occurs.

When queried, the value of this parameter indicates whether the device is currently mounted. `Mounted` should be queried immediately after it is set, to obtain the result of the attempted mount or dismount.

**Removable** boolean (Read-only) A flag specifying whether the device supports removable media. Depending on how the removable-media device operates, setting the value of `Mounted` to `false` for such a device will either eject the medium or allow the medium to be removed. Once the medium has been removed, the device cannot be mounted again until the medium is reinserted.

**Writeable** boolean (Read-only except during a mount) A flag specifying whether the files on the device can be opened for write access (or, if the device is not mounted, whether the device will support writeable media). If the medium is write-protected, `Writeable` is `false`. This parameter can be set only during a mount—that is, at the same time that the `Mounted` parameter is being set to `true`—and only if the medium is not write-protected.

**Searchable** boolean A flag specifying whether the device participates in searches for a file when a file name is supplied that does not specify a device (see PLR3, Section 3.8.2).

**Note:** On some products, devices that support removable media will initially have `Searchable` set to `false`. `Searchable` must be explicitly set to `true` on such devices to enable them to participate in device searches.

**SearchOrder** integer The priority (0 or greater) at which the device participates in file searches as indicated by a value of `true` for the `Searchable` parameter. The lower the value, the higher the priority. This parameter is ignored if `Searchable` is `false`.

**BlockSize** integer (Read-only) The number of bytes in a block on the formatted device. For a disk using the Adobe file system, `BlockSize` is 1024. This parameter determines the unit for the values of the `Free`, `PhysicalSize`, and `LogicalSize` parameters.

**Free** integer (Read-only) The number of blocks of free space available on the medium in the device (where the block size is indicated by the `BlockSize` parameter). `Free` is 0 if the medium is completely full or if the device is not mounted.
Device Parameters

**PhysicalSize** integer

(Read-only) The number of blocks of medium in the device (where the block size is indicated by the `BlockSize` parameter). This is the maximum allowable size of the file system; the exact size is determined by the `LogicalSize` parameter. `PhysicalSize` is 0 if the device is not mounted.

**LogicalSize** integer

The number of blocks allocated to the file system (where the block size is indicated by the `BlockSize` parameter). When queried, `LogicalSize` is 0 if the device is not mounted.

When set, this parameter specifies the number of blocks to be allocated to the file system when it is created in response to the `InitializeAction` parameter. The value of `LogicalSize` must not exceed the value of `PhysicalSize`. If `LogicalSize` is 0, `InitializeAction` uses a default size that is normally the same as the value of `PhysicalSize`.

If `LogicalSize` is set with a certain value and (as specified by the value of `InitializeAction`) the medium in the device is reformatted before the file system is created, a subsequent query of `LogicalSize` should return the value that was set. However, if `LogicalSize` is queried before the medium is reformatted, it may return the current size rather than the value that was set for it.

**InitializeAction** integer

A code specifying an action for initializing the device:

- 0 No action. This value is returned when the parameter is queried.
- 1 Deletes the current file system (if any) and creates one having the size specified by the `LogicalSize` parameter. (The medium is assumed to have been formatted already.) The device must first be mounted; otherwise, an `ioerror` will occur.
- 2 Reformats the entire medium before creating a new file system of size `LogicalSize`. The `Interleave` parameter also plays a role in how the medium is formatted; see the description of `Interleave` for details.
- ≥3 Has the same effect as the value 2 and also carries out product-dependent actions, which typically consist of preformatting the disk and running integrity tests before creating the file system. Some devices can have additional parameters that serve as arguments to `InitializeAction`.

**Interleave** integer

The interleave number, $n$, specifying $n$-to-$1$ interleaving. Interleaving arranges logically contiguous sectors on the disk in the most efficient way for the system using that disk. This parameter is used only when the medium is being formatted in response to the `InitializeAction` parameter.

For example, assume there are 16 sectors going around a single track on a disk. If the first sector is logical sector number 1, the second 2, the third 3, and so on, the value of `Interleave` is 1 (1-to-1 interleaving). In this case, the
system must be very fast to be able to take data from the disk, one sector im-
mmediately after another. If the system fails to consume the first sector in time
for the second sector, it has to wait an entire revolution of the disk to get the
next sector. This can result in very poor performance.

If the first sector is logical sector number 1, the third 2, the fifth 3, and so on,
the system will need to be able to consume the current sector while the head
skips over a sector in time for the next logical sector. In this case, the value of
Interleave is 2 (2-to-1 interleaving). The sectors in between are used for high-
er logical numbers, and it takes a minimum of two revolutions to get the data
for an entire track off the disk. In this example, the second physical sector on
the disk would be between logical sectors 1 and 2, and would be logical sector
9.

Similarly, with an Interleave value of 3, the first sector is logical sector num-
ber 1, the fourth 2, and so on.

Normally, Interleave should be set to a value that allows the software to use
the data during the time between sectors, but not waste any time. It is diffi-
cult to determine what the proper value is, and the value depends greatly on
the job accessing the disk. For drives that provide buffering for a full track of
data, 1-to-1 interleaving is almost always most efficient.

**PrepareAction** integer

A code specifying an action to prepare the underlying file system for a specific
purpose:

0  No action.

1  A product-specific action that loads system files. In one case, these
files support older versions of Adobe Japanese typefaces and, on a
writeable file system, enable older versions of the Japanese Font
Downloader utility to work correctly.

If InitializeAction and PrepareAction are set in the same invocation of
setdevparams, the actions performed in response to InitializeAction precede
those performed in response to PrepareAction.

**TABLE 3.33 Device parameters in the %cartridge% (or %cartridgen%) and %rom% (or %romn%) parameter sets**

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td><em>(Read-only)</em> The parameter set type; must be FileSystem.</td>
</tr>
</tbody>
</table>
| CartridgeType | integer | *(Read-only)* The category classification of the cartridge, from a registry main-
                                                          tained by Adobe. |
CartridgeID  integer  (Read-only) A number that uniquely identifies this cartridge on a product. This parameter is used by the interpreter to determine whether a cartridge has been removed from a slot and a different cartridge inserted.

HasNames  boolean  (Read-only) See Table 3.32 above.
Mounted  boolean  See Table 3.32 above.
Removable  boolean  (Read-only) See Table 3.32 above.
Writeable  boolean  (Read-only except during a mount) See Table 3.32 above.
Searchable  boolean  See Table 3.32 above.
SearchOrder  integer  See Table 3.32 above.
BlockSize  integer  (Read-only) See Table 3.32 above. The value of BlockSize for these parameters is typically 1.
Free  integer  (Read-only) See Table 3.32 above.
PhysicalSize  integer  (Read-only) See Table 3.32 above.
LogicalSize  integer  See Table 3.32 above.
InitializeAction  integer  A code specifying an action for initializing the device. The valid values for writeable cartridges are listed below. (Setting InitializeAction for a read-only cartridge has no effect.)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No action. This value is returned when the parameter is queried.</td>
</tr>
<tr>
<td>1</td>
<td>Reformats the entire medium and then creates a new file system using the full size of the cartridge.</td>
</tr>
</tbody>
</table>

### TABLE 3.34 Device parameters in the %ram% (or %ramn%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be FileSystem.</td>
</tr>
<tr>
<td>HasNames</td>
<td>boolean</td>
<td>(Read-only) A flag specifying whether the device supports named files; always true.</td>
</tr>
<tr>
<td>Mounted</td>
<td>boolean</td>
<td>(Read-only) A flag specifying whether the device is mounted; always true.</td>
</tr>
<tr>
<td>Removable</td>
<td>boolean</td>
<td>(Read-only) A flag specifying whether the device supports removable media; always false.</td>
</tr>
<tr>
<td>Writeable</td>
<td>boolean</td>
<td>(Read-only) A flag specifying whether the files on the device can be opened for write access; always true.</td>
</tr>
<tr>
<td>Searchable</td>
<td>boolean</td>
<td>(Read-only) See Table 3.32 on page 61. This parameter is always false.</td>
</tr>
</tbody>
</table>
SearchOrder integer (Read-only) See Table 3.32 on page 61. This parameter (whose value is always 100) is ignored in this parameter set, since Searchable is always false.

BlockSize integer (Read-only) See Table 3.32 on page 61. The value of BlockSize for this parameter set is typically 1.

Free integer (Read-only) The number of blocks of free space available on the medium for the device (where the block size is indicated by the BlockSize parameter). Free is 0 if the medium is completely full.

PhysicalSize integer (Read-only) See Table 3.32 on page 61.

LogicalSize integer See Table 3.32 on page 61.

InitializeAction integer A code specifying an action for initializing the device:

0 No action. This value is returned when the parameter is queried.
1 Reformats the entire medium and then creates a new file system of size LogicalSize.

### TABLE 3.35 Device parameters in the %os% (or %osX%) parameter set

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) The parameter set type; must be FileSystem.</td>
</tr>
<tr>
<td>HasNames</td>
<td>boolean</td>
<td>(Read-only) A flag specifying whether the device supports named files; always true.</td>
</tr>
<tr>
<td>Mounted</td>
<td>boolean</td>
<td>(Read-only) A flag specifying whether the device is mounted; always true.</td>
</tr>
<tr>
<td>Removable</td>
<td>boolean</td>
<td>(Read-only) A flag specifying whether the device supports removable media; always false.</td>
</tr>
<tr>
<td>Writeable</td>
<td>boolean</td>
<td>(Read-only) A flag specifying whether the files on the device can be opened for write access; always true.</td>
</tr>
<tr>
<td>Searchable</td>
<td>boolean</td>
<td>(Read-only) See Table 3.32 on page 61.</td>
</tr>
<tr>
<td>SearchOrder</td>
<td>integer</td>
<td>See Table 3.32 on page 61. The value of this parameter is initially 2.</td>
</tr>
<tr>
<td>BlockSize</td>
<td>integer</td>
<td>(Read-only) The number of bytes in a block (for the partition dedicated to PostScript). This parameter is set by the environment, not by PostScript, and is typically 1024.</td>
</tr>
<tr>
<td>Free</td>
<td>integer</td>
<td>(Read-only) The number of blocks of free space available in the PostScript partition (where the block size is indicated by the BlockSize parameter). Free is 0 if the partition is completely full.</td>
</tr>
</tbody>
</table>
**Device Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhysicalSize</td>
<td>integer</td>
<td>(Read-only) The number of blocks in the PostScript partition (where the block size is indicated by the BlockSize parameter).</td>
</tr>
<tr>
<td>LogicalSize</td>
<td>integer</td>
<td>(Read-only) The number of blocks allocated to the file system (where the block size is indicated by the BlockSize parameter). In this parameter set, the values of LogicalSize and PhysicalSize will always be the same.</td>
</tr>
<tr>
<td>InitializeAction</td>
<td>integer</td>
<td>(Read-only) A code specifying an action for initializing the device; always 0, indicating no action.</td>
</tr>
</tbody>
</table>
CHAPTER 4

Resources

THIS CHAPTER SUPPLEMENTS Section 3.9 of the PostScript Language Reference, Third Edition. It provides information about additional named resources that are relatively device-specific—those that describe features unique to a particular output device or limited to only a few devices.

4.1 Regular Resource Categories

Regular resources are those whose instances are ordinary useful objects, such as font or halftone dictionaries. Most categories of regular resources are described in PLR3, but a few of them are product-dependent and are described in this Supplement instead. These include the following:

- Instances of ControlLanguage are dictionaries that describe the control languages available in a product. A control language is a means for controlling product features, such as default configuration and status reporting.

- Instances of PDL are dictionaries that describe the page description language interpreters available in a product. This category supersedes the implicit resource category Emulator, since its instances provide a more complete description of each interpreter (or emulator).

- Instances of Localization are dictionaries that describe the natural languages (such as English, Japanese, or German) supported by a product—that is, the language of its user interface.

- Instances of HWOptions are strings that describe the special hardware options present in a product.

These resource categories are described further in the following sections.
4.1.1 ControlLanguage

The ControlLanguage resource category enables an application or printer driver to determine what control languages are available on a given output device. A control language determines how the environment and device parameters are configured, how jobs are identified, and the format of device-generated messages on the back channel. Each instance of the ControlLanguage resource category is a dictionary whose contents describe one of the control languages supported by the device. Table 4.1 shows the entries in such a dictionary.

| TABLE 4.1 Entries in an instance of the ControlLanguage resource category |
|-----------------------------|-----------------|----------------------------------|
| KEY                        | TYPE            | VALUE                           |
| Selector                   | name            | (Required) A name object that can be used as the value of the PrinterControl device parameter in any parameter set of type Communications (see Table 3.3 on page 29). |
| LanguageFamily             | string          | (Required) The name of the control language, such as PostScript or PJL. |
| LanguageLevel              | string          | (Optional) A string identifying the level of the control language specified by LanguageFamily that is implemented on this device. This can be a level identifier, such as 2 (LanguageLevel 2) for the PostScript language family; for a language family such as PJL, which has no such level designations, the value is an empty string. |
| LanguageVersion            | string          | (Optional) A string identifying the version of the control language. This is typically a microcode version identifier, such as 2015.101; it can also be a product name, such as LaserJet 4si. |

4.1.2 PDL

The PDL resource category enables an application or printer driver to determine what page description language (PDL) interpreters are available on a given output device. Each instance of this category is a dictionary whose contents describe one of the page description languages supported by the device. Table 4.2 shows entries that are typically present in such a dictionary. Additional entries may be added to further characterize the page description language.
4.1.3 Localization

The **Localization** resource category enables an application or printer driver to determine what natural languages (such as English, Japanese, or German) are available on a given output device. Each instance of this category is a dictionary that identifies a particular language, national dialect, and character set that are supported by the device. Table 4.3 shows the entries in such a dictionary. Note that these are the same as the entries in the %Console% parameter set; see Table 3.28 on page 57 for further information.

### TABLE 4.3 Entries in an instance of the Localization resource category

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>name</td>
<td>A name object specifying the language with a 2-character language code from the ISO 639 standard. See Tables 3.28 and 3.29 on pages 57 and 58 for details.</td>
</tr>
<tr>
<td>Country</td>
<td>name</td>
<td>A name object that indirectly specifies the dialect of the language by specifying the country in which the dialect is used. The country is indicated by a 2-character country code from the ISO 3166 standard. See Tables 3.28 and 3.30 on pages 57 and 59 for details.</td>
</tr>
</tbody>
</table>
CharSet name A name object specifying either a standard character set or a vendor-specific character set. See Table 3.28 on page 57 for details.

Each Localization instance represents a particular combination of Language, Country, and CharSet values. In general, any given device will support only a sparse subset of all possible combinations. For example, the entire Localization category might consist of the following instances:

```
<< /Language /EN
    /Country /US
    /CharSet /ISO-646-ISV
>>
```

```
<< /Language /JA
    /Country null
    /CharSet /JIS-
>>
```

where the code EN denotes English, JA denotes Japanese, US denotes the United States dialect of English, and the null value indicates that no specific dialect is identified for Japanese.

Each Localization instance has a unique name. Although no specific naming scheme is enforced, the following guidelines are recommended:

• Instances can be named by concatenating the values of the three dictionary entries, separated by hyphens. For example, the two instances shown above would be named

  /EN-US-ISO-646-ISV

and

  /JA-JIS-

respectively. This approach yields a name indicative of the dictionary’s contents.

• An alternative is to provide descriptive names for the instances. For example, the two instances shown above might be named

  /AmericanEnglish

and
Either naming scheme provides a simple way for a PostScript application or printer driver to specify the language to be used with the device. Using the second naming scheme, for example, the following PostScript code changes the language to Japanese:

\[
\text{% Set the console to use the Japanese language} \\
(\text{%Console%}) \\
\text{/Japanese /Localization findresource} \\
\text{setdevparams}
\]

**Note:** If the system parameter `SystemParamsPassword` is set, it is also necessary to include a valid `Password` entry in the `Localization` resource or run this code as an unencapsulated system administrator job; for details, see PLR3, Section C.3.1.

### 4.1.4 HWOptions

Instances of the `HWOptions` resource category are strings identifying any special hardware options that may be present on a given output device. This category is available if needed to address the special requirements of a particular device; however, because the exact details are device-dependent, no specific instances of the category are officially defined at the time of publication.

### 4.2 Implicit Resource Categories

Implicit resources represent some built-in capability of the PostScript interpreter. For example, instances of the `Filter` category are filter names, such as `ASCII85Decode` and `CCITTFaxDecode`, that are passed directly to the `filter` operator. Table 4.4 lists implicit resources that are not described in PLR3.

<table>
<thead>
<tr>
<th>CATEGORY NAME</th>
<th>INSTANCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>IODevice</td>
<td>%Serial%</td>
</tr>
<tr>
<td></td>
<td>%Serial_NV%</td>
</tr>
<tr>
<td></td>
<td>%Serial_Pending%</td>
</tr>
<tr>
<td></td>
<td>%Parallel%</td>
</tr>
<tr>
<td></td>
<td>%Parallel_NV%</td>
</tr>
<tr>
<td></td>
<td>%Parallel_Pending%</td>
</tr>
<tr>
<td></td>
<td>%ScsiComm%</td>
</tr>
<tr>
<td></td>
<td>%ScsiComm_NV%</td>
</tr>
<tr>
<td></td>
<td>%ScsiComm_Pending%</td>
</tr>
<tr>
<td></td>
<td>%LocalTalk%</td>
</tr>
<tr>
<td></td>
<td>%LocalTalk_NV%</td>
</tr>
<tr>
<td></td>
<td>%LocalTalk_Pending%</td>
</tr>
<tr>
<td></td>
<td>%EtherTalk%</td>
</tr>
<tr>
<td></td>
<td>%EtherTalk_NV%</td>
</tr>
<tr>
<td></td>
<td>%EtherTalk_Pending%</td>
</tr>
<tr>
<td></td>
<td>%TokenTalk%</td>
</tr>
<tr>
<td></td>
<td>%TokenTalk_NV%</td>
</tr>
<tr>
<td></td>
<td>%TokenTalk_Pending%</td>
</tr>
</tbody>
</table>
Instances of the **IODevice** category refer to device parameter sets described in Chapter 3. The **Emulator** resource category has been superseded by the **PDL** resource in LanguageLevel 3 and may not be present on all products.

**GIFDecode** and **PNGDecode** are optional filters that are used only as part of the implementation of Web printing (printing HTML content from the World Wide Web). PostScript programs should not invoke these filters.
CHAPTER 5

Other Extensions

THIS CHAPTER PROVIDES information about miscellaneous extensions to the PostScript language that are not described in the *PostScript Language Reference, Third Edition*. These include new and extended font types and extensions to the contents of image dictionaries and trapping parameter dictionaries. The chapter also provides guidelines for synchronizing PostScript color rendering dictionaries (CRDs) with device color profiles as defined by the International Color Consortium (ICC).

5.1 Font Extensions

This section supplements Chapter 5 of the *PostScript Language Reference, Third Edition*. It describes:

- The `%fontset%` fictitious file system, the compatibility mechanism that ensures that all programs will work with fonts defined in `FontSet` resource instances
- A new kind of Type 0 CIDFont: a CFF CIDFont, defined in `FontSet` resources rather than in a CIDFont file
- Extensions to the Type 2 CIDFont dictionary to accommodate incremental downloading of CJK TrueType fonts

5.1.1 `%fontset%` Fictitious File System

As described in *PLR3*, Section 5.8.1, Type 2 (Compact Font Format, or CFF) and Type 14 (Chameleon) fonts are integrated into the PostScript font model in a way that is compatible with both LanguageLevel 1 (file system) and LanguageLevel 2 (resource) methods for accessing fonts. There exists a fictitious file system named
Other Extensions

CHAPTER 5

%fontset%, whose “files” are the constituent fonts in all available FontSet resource instances. This file system has the following characteristics and behavior:

- It is read-only.
- Its “files” appears to have names of the form fonts/Palatino. The prefix is the constant string fonts/, which is the recommended value of the FontResourceDir system parameter; this is followed by the name of a font in some FontSet resource instance.
- It is typically searchable—that is, the default value of its Searchable parameter is usually true. This means that it is among the file systems considered when a file name is supplied that does not specify a device (see PLR3, Section 3.8.2).
- Font lookup operations will tend to cause all FontSet resource instances to be loaded into virtual memory (VM) as a side effect. A FontSet dictionary takes minimal VM (a few hundred bytes) if loaded from a positionable file. Furthermore, FontSet instances are expected to be few in number. (Most printers will have at most three—one for all CFF fonts and one each for a Chameleon master font and its descriptor database.)
- File systems are searched in order of their SearchOrder device parameter. Thus, the %fontset% file system will be consulted before or after other file systems, such as cartridges or disks, according to their relative SearchOrder values. Specifying that %fontset% be searched first will yield the best performance for fonts present in the FontSet dictionaries (since no other file systems need to be consulted), but it will preclude overriding a FontSet font by installing a font with the same name on disk.
- The Searchable and SearchOrder parameters of the %fontset% file system can be changed, with the setdevparams operator.

Opening a file with

(%fontset%fonts/Palatino) (r) file

or (if %fontset% is the file system selected by the search rule) with

(fonts/Palatino) (r) file

causes the FontSet resource instances to be enumerated until one containing the named font is found. The resource instances are loaded into VM as a side effect. A fictitious file is then fabricated and made available for reading. When this ficti-
tious file is executed, it causes the font dictionary to be built and defined as an instance of the Font resource category. The contents of the file are undocumented and subject to change.

*Note:* The fictitious file does not have any document structuring conventions (DSC); consequently, a Font resource status operation will not find a %%VMusage: comment and will return a size of −1.

The resulting font dictionary has a **FontType** value of 2 for CFF or 14 for Chameleon, plus all the required entries for a Type 1 font dictionary (as defined in PLR3, Tables 5.2, 5.3, and 5.4 on pp. 324–327), except that it does not have a **Private** entry. It may also contain additional entries that are undocumented and subject to change.

A PostScript program can copy the font dictionary and insert or modify entries as specified in the aforementioned tables. These modifications have the same effects for Type 2 and Type 14 fonts as for Type 1 fonts.

Executing the **status** operator for a valid file name in the %fontset% file system yields four meaningless numbers followed by *true* if the named font exists in any font set; otherwise it yields *false*.

### 5.1.2 CFF CIDFonts

Type 2 (Compact Font Format, or CFF) base fonts and Type 0 CIDFonts are both based on Type 1 font technology. There is also a special kind of Type 0 CIDFont: a CFF CIDFont. Other Type 0 CIDFonts are defined in CIDFont files, but CFF CIDFonts, like Type 2 base fonts, are defined in FontSet resources: they are integrated into the PostScript font model via the %fontset% fictitious file system, as described above.

Like other Type 0 CIDFonts, a CFF CIDFont is characterized by **CIDFontType** and **FontType** values of 0 and 9, respectively. It is further characterized by a **FontType** value of 2 in each **FDArray** subdictionary.

For more information, see Section 5.8.1 (pp. 343–346) and “Type 0 CIDFonts” (pp. 371–376) in PLR3.
5.1.3 Incremental Downloading of CJK TrueType CIDFonts

As described in PLR3, Section 5.11, CID-keyed fonts provide great flexibility for representing text in writing systems for languages with large character sets, such as Chinese, Japanese, and Korean (CJK). CJK TrueType fonts can be downloaded as CID-keyed fonts containing Type 2 CIDFonts, which are the analog of Type 42 (TrueType) base fonts.

Applications or drivers usually download CJK fonts as CIDFonts into PostScript VM or install them on a printer disk. Some drivers, however, particularly Windows® drivers, need to be able to download fonts incrementally or as subsets to minimize the transmission cost of a print job. To accommodate incremental downloading of CJK TrueType fonts, entries in the Type 2 CIDFont dictionary have been extended as follows:

- The definition of the CIDMap entry can be a dictionary or an integer as well as a string or an array, as described in the next section.
- The definition of the GlyphDirectory entry has been extended, and an optional MetricsCount entry added, to allow the inclusion of glyph metric information in GlyphDirectory, as described in the section “GlyphDirectory and Metrics-Count” on page 79.

New CIDMap Types

The value of CIDMap can be not only a string or an array of strings but also a dictionary or an integer.

CIDMap can be defined as a dictionary that maps CIDs directly to glyph indices, as follows:

```
/CIDMap
  << CID0 GI0
  ...
  CIDn GI
>>
```

where CID0…CIDn are the CIDs—that is, 0 through (CIDCount – 1)—and GI0…GIn are the corresponding glyph indices. The glyph indices are then used to locate the corresponding glyph descriptions in GlyphDirectory. The CIDMap dictionary can be downloaded incrementally as glyph descriptions are needed.
If a driver can map CIDs to glyph indices directly, it is no longer necessary to download CIDMap at all; this is where defining CIDMap as an integer comes in. The mapping relationship can be defined so that there is a constant integer offset from the identity mapping, as follows:

/CIDMap integer def

With this definition, each CID maps to a glyph index that is the sum of the CID and integer. This allows fonts having many glyphs to be represented as several separate CIDFonts. A typical Korean font, for example, can contain 22,000 glyphs, which would be accessed using glyph indices 0 through 21,999. Because of PostScript limitations, such a font would have to be represented as two separate CIDFonts, for which integer would be 0 and 16,000, respectively. The CIDMap for the first CIDFont would be defined as

/CIDMap 0 def

so that the CID 0 maps to the glyph index 0, the CID 1 to the glyph index 1, and so on, up to 15,999. The CIDMap for the second CIDFont would be defined as

/CIDMap 16000 def

so that the CID 0 maps to the glyph index 16,000, the CID 1 to the glyph index 16,001, and so on, until the CID 5999 maps to 21,999.

GlyphDirectory and MetricsCount

To facilitate incremental downloading of Type 2 CIDFonts, glyph metrics may optionally be defined in the GlyphDirectory entry in the Type 2 CIDFont dictionary (as an alternative to defining them in the TrueType “hmtx” and “vmtx” tables embedded in the sfnts array). Including glyph metric information in GlyphDirectory for a subset of characters avoids the need to download large tables, such as the “hmtx” and “vmtx” tables. Whether the metrics are included in GlyphDirectory, and which writing mode or modes they apply to, is defined in the MetricsCount entry in the Type 2 CIDFont dictionary.

The optional MetricsCount entry—an integer with a default value of 0—specifies the location of the glyph metric information. MetricsCount may have a value of 0, 2, or 4, indicating that the first 0, 4, or 8 bytes of the glyph description contain the
metric information—that is, the glyph description begins $\text{MetricsCount} \times 2$ bytes from the beginning of the string.

As described in PLR3, Section 5.4, the glyph metric information includes:

- The glyph’s width (the distance the current point moves when the glyph is shown, not the dimensions of the glyph outline)
- The glyph’s left sidebearing

These are the general terms used for these metrics, independent of the writing mode (that is, whether WMode is 0 for horizontal or 1 for vertical); however, for writing mode 1, the metrics are actually the glyph’s height and top sidebearing.

If $\text{MetricsCount}$ is 0:

- Glyph metric information for writing mode 0 is obtained from the “hmtx” table or the Metrics dictionary.
- Glyph metric information for writing mode 1 is obtained from the “vmtx” table, the Metrics2 dictionary, or CDevProc.

If $\text{MetricsCount}$ is 2:

- Glyph metric information for writing mode 0 is obtained from the width and left sidebearing data at the beginning of the glyph description defined in GlyphDirectory.
- Glyph metric information for writing mode 1 is obtained from the “vmtx” table, the Metrics2 dictionary, or CDevProc.

If $\text{MetricsCount}$ is 4:

- Glyph metric information for writing mode 0 is obtained from bytes 4 and 5 (for the width) and bytes 6 and 7 (for the left sidebearing) at the beginning of the glyph description defined in GlyphDirectory.
- Glyph metric information for writing mode 1 is obtained from bytes 0 and 1 (for the height) and bytes 2 and 3 (for the top sidebearing) at the beginning of the glyph description defined in GlyphDirectory.

Table 5.1 summarizes the meanings of the different values of $\text{MetricsCount}$. 
### TABLE 5.1 Meanings of MetricsCount values

<table>
<thead>
<tr>
<th>MetricsCount VALUE</th>
<th>BYTES AT BEGINNING OF GLYPH DESCRIPTION</th>
<th>SOURCE OF METRICS FOR WMode 0 (HORIZONTAL)</th>
<th>SOURCE OF METRICS FOR WMode 1 (VERTICAL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>“hmtx” table or Metrics</td>
<td>“vmtx” table, Metrics2, or CDevProc</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Bytes at beginning of glyph description:</td>
<td>Bytes at beginning of glyph description:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bytes 0, 1: Width</td>
<td>Bytes 0, 1: Height</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bytes 2, 3: Left sidebearing</td>
<td>Bytes 2, 3: Top sidebearing</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>Bytes at beginning of glyph description:</td>
<td>Bytes at beginning of glyph description:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bytes 4, 5: Width</td>
<td>Bytes 0, 1: Height</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bytes 6, 7: Left sidebearing</td>
<td>Bytes 2, 3: Top sidebearing</td>
</tr>
</tbody>
</table>

### 5.2 Interpolation Details Dictionary

A new optional entry (see Table 5.2) has been added to all image dictionaries, regardless of type. Image dictionaries are described in PLR3, Section 4.10.5.

### TABLE 5.2 Additional entry in all image dictionaries

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>SEMANTICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterpolationDetails</td>
<td>dictionary</td>
<td>(Optional) A dictionary containing device-specific information related to image interpolation. Like all details dictionaries, this dictionary should include a Type entry whose integer value determines the remaining structure and contents of the dictionary (see Section 2.1, “Details Dictionaries,” as well as PLR3, Section 6.1.2). This parameter is ignored if the value of Interpolate is false.</td>
</tr>
</tbody>
</table>

### 5.3 Imagemask Trapping Parameter

A new optional trapping parameter (see Table 5.3) has been added to the trapping parameter dictionary. Trapping parameter dictionaries are described in PLR3, Section 6.3.3.
**TABLE 5.3 Additional trapping parameter**

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>SEMANTICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImagemaskTrapping</td>
<td>boolean</td>
<td><em>(Optional)</em> A flag specifying whether trapping should be performed between stencil masks and the background against which they are painted. When this flag is false, stencil masks are painted within the given trapping zone with an implicit colorant type of OpaqueIgnore (see ColorantType in PLR3, p. 444). That is, no traps are generated between the stencil mask and other objects that overlay it or that it overlays. Note, however, that objects that overlay the stencil mask and overlap each other will be trapped to each other. Default value: true.</td>
</tr>
</tbody>
</table>

When a page description contains a stencil mask to be drawn with the image-mask operator (PLR3, pp. 302–303 and 608–610), it frequently is not appropriate to trap the stencil mask against the background on which it is painted. Particularly when two or more stencil masks overlap or when the area covered by a stencil mask is small, trapping can produce undesirable smear effects. The trapping parameter ImagemaskTrapping can be used to enable or disable such trapping.

### 5.4 Color Rendering Dictionaries and ICC Profiles

The International Color Consortium (ICC) has defined a standard format, known as an ICC profile, for describing the color properties of an output device. ICC profiles can contain the same information as PostScript color rendering dictionaries (CRDs), and can be translated to or from CRDs in a straightforward way. Color rendering dictionaries are described in PLR3, Section 7.1. This section describes how to coordinate CRDs on the output device with ICC profiles on the host computer, in order to reduce the memory requirements and file transmission times for color transformations. Such equivalent CRDs and profiles on the device and host are known as companions.

Every CRD should include a CreationDate entry (PLR3, p. 468), whose value is a string indicating the date and time at which the CRD was created or most recently modified. This value should be coordinated with the calibrationDateTimeTag entry of any companion profile. (Even if no companion profile is present on the host, the CreationDate entry should still be included in the CRD.) The format of the CreationDate string should match that of the calibrationDateTimeTag entry.
in the ICC profile, as defined in the document *ICC Profile Format Specification* (see the Bibliography):

\[ YYYYMMDDHHmmSSOH' \]

where

\[ YYYY \] is the year
\[ MM \] is the month
\[ DD \] is the day (01–31)
\[ HH \] is the hour (00–23)
\[ mm \] is the minute (00–59)
\[ SS \] is the second (00–59)
\[ O \] is the relationship of local time to Universal Time (UT); see below
\[ HH' \] is the absolute value of the offset from UT in hours (00–23)
\[ mm' \] is the absolute value of the offset from UT in minutes (00–59)

Fields after the year are optional. The default values for \[ MM \] and \[ DD \] are both 01; all other numerical fields default to zero values. A plus sign (+) as the value of the \[ O \] field indicates that local time is later than UT, a minus sign (-) indicates that local time is earlier than UT, and the letter Z indicates that local time is equal to UT. If no UT information is specified, the relationship of the specified time to UT is considered to be unknown. Whether or not the time zone is known, the date should be specified based on local time.

Profiles should be extended with the optional ICC entry \texttt{crdInfoTag}, containing the PostScript product name to which this profile corresponds and the names of the companion CRDs. Note that a single profile can have multiple companion CRDs, corresponding to different rendering intents (see *PLR3*, Section 7.1.3). Table 5.4 gives the format of the \texttt{crdInfoTag} entry.

<table>
<thead>
<tr>
<th>TABLE 5.4 Format of \texttt{crdInfoTag}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BYTES</strong></td>
</tr>
<tr>
<td>0–3</td>
</tr>
<tr>
<td>4–7</td>
</tr>
<tr>
<td>8–11</td>
</tr>
<tr>
<td>12–(( M - 1 ))</td>
</tr>
</tbody>
</table>
CRDs are stored on the output device as instances of resource category **Color-Rendering** with the names specified in the profile’s `crdInfoTag` entry. If no companion CRD exists for a given rendering intent, the corresponding character count is zero and there is no CRD name string.

The CRD’s `CreationDate` entry and the ICC profile’s `crdInfoTag` entry can be coordinated differently depending on whether bidirectional communications are available between the host and the output device and whether the CRD was supplied with the output device or was downloaded from a host in the field. Bidirectional communication allows the output device to be queried to determine the availability of a given CRD and its associated `CreationDate` entry.

In the absence of bidirectional communications, the list of device-resident CRDs and their `CreationDate` entries is available through the output device’s PostScript printer description (PPD) and (on some platforms) the host profile registry. PPDs currently contain the names of the CRDs that ship with an output device; the PPD format will be extended to contain the `CreationDate` entry for each CRD as well. The registry should be updated whenever CRDs are downloaded or created in the field.
To determine whether a suitable CRD is available on the output device, a PostScript driver can do the following:

1. Obtain the device's PostScript product name, either from the PPD or via the PostScript `product` operator, and compare it with the product names given in the `crdInfoTag` entries of all available ICC profiles on the host. This comparison limits the selection of profiles to only those pertaining to the given output device.

2. For each profile whose product name matches that of the device, look in the `crdInfoTag` entry for the CRD name corresponding to the required rendering intent and check whether the device has a `ColorRendering` resource with that name.

3. If a resource with the specified name exists, compare its `CreationDate` entry with the profile's `calibrationDateTimeTag` entry. If the two match, the resource contains the required CRD.

If no suitable CRD is found on the device, the driver can download one from an ICC profile on the host. Ordinarily, such a downloaded CRD will not be available for subsequent jobs, but the driver can make it persistent by giving it an appropriate `CreationDate` entry, updating the registry, and updating the profile's `crdInfoTag` entry to refer to this CRD.

If a CRD on the output device has no corresponding profile on the host, the driver can coordinate it with a companion profile either by copying its `CreationDate` entry to the profile or by updating the CRD and registry from the profile's `calibrationDateTimeTag` entry. In either case, the profile's `crdInfoTag` entry must be filled in correctly. Note also that in this case the CRD updates may not persist across power cycles of the output device; such power cycles should result in the registry being updated.
MOST POSTSCRIPT IMPLEMENTATIONS provide a standard dictionary named `statusdict` to hold product-specific operators and other data whose names and values vary from one product to another (and sometimes even from version to version of the same product). Information in `statusdict` is associated with unique features of a product that cannot be accessed in any standard way. Although the contents of `statusdict` are product-dependent, an attempt has been made to maintain a consistent specification for common features that are found in more than one product.

In LanguageLevel 1, `statusdict` includes a variety of operators to select print-engine features, set communication parameters, and control other product-specific aspects of the PostScript interpreter’s operating environment. In LanguageLevels 2 and 3, most of these functions have been subsumed by standard operators, such as `setsystemparams`, `setdevparams`, and `setpagedevice`. However, LanguageLevel 2 and LanguageLevel 3 implementations continue to provide a set of compatibility operators and related data values corresponding to the older features, in order to maintain compatibility with existing LanguageLevel 1 software that might depend on their presence. These compatibility operators and data values are the subject of this chapter.

Unlike the standard PostScript operators described in the PostScript Language Reference, Third Edition (which reside in the standard dictionary `systemdict`), the majority of compatibility operators and data values are defined in `statusdict` (though a few of them are in `systemdict` or `userdict` instead). Because the availability of specific operators and data values is product-dependent, their continued use beyond LanguageLevel 1 is not recommended. The appropriate LanguageLevel 2 and LanguageLevel 3 operators should be used instead.
Note: The behavior of many of the compatibility operators is defined in terms of setting or reading corresponding system, user, device, or page device parameters. A given compatibility operator typically will be defined only if its associated parameter exists; however, this is not invariably the case.

Note also that some of the compatibility “operators” are actually implemented as PostScript procedures rather than true operators. These procedures are designed to behave exactly the same as the original LanguageLevel 1 operators; for example, they mimic the exact error behavior of the original operators, even where it differs from that of the more general LanguageLevel 2 and LanguageLevel 3 operators that have replaced them. The distinction between procedures and true operators is thus immaterial from the viewpoint of the calling program.

This chapter is divided into two sections:

- Section 6.1 gives a summary of the operators and data values, organized into groups of related functions. The summary is intended to help locate the operators needed to perform specific tasks.

- Section 6.2 gives detailed descriptions of all operators and data values, organized alphabetically by name.

### 6.1 Compatibility Operator Summary

#### System Configuration

<table>
<thead>
<tr>
<th>Type</th>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>setprintername</td>
<td>Set device name</td>
</tr>
<tr>
<td>string</td>
<td>printername substring</td>
<td>Return device name string</td>
</tr>
<tr>
<td>string</td>
<td>product string</td>
<td>Value of product name string</td>
</tr>
<tr>
<td>int</td>
<td>revision</td>
<td>Value of product revision level</td>
</tr>
<tr>
<td>int</td>
<td>buildtime</td>
<td>Value of interpreter time stamp</td>
</tr>
<tr>
<td>bool</td>
<td>byteorder bool</td>
<td>Value of native byte order</td>
</tr>
<tr>
<td>string</td>
<td>realformat string</td>
<td>Value of native real number format</td>
</tr>
<tr>
<td>int</td>
<td>ramsize</td>
<td>Return available RAM size</td>
</tr>
<tr>
<td>int</td>
<td>pagecount</td>
<td>Return device page count</td>
</tr>
<tr>
<td>bool</td>
<td>setdostart bool</td>
<td>Set system startup mode</td>
</tr>
<tr>
<td>bool</td>
<td>setdostartpage</td>
<td>Set start page flag</td>
</tr>
<tr>
<td>bool</td>
<td>setstartpage bool</td>
<td>Return start page flag</td>
</tr>
<tr>
<td>bool</td>
<td>setdoprinters bool</td>
<td>Set error printing mode</td>
</tr>
<tr>
<td>bool</td>
<td>doprinters bool</td>
<td>Return error printing mode</td>
</tr>
</tbody>
</table>
Job Control

- **jobname** string  
  Value of job name string

- **checkpassword** bool  
  Check password string for startjob or exitserver

- **checkpassword** bool  
  Check password integer for startjob or exitserver

- **setdefaulttimeouts**  
  Set default job, manual feed, and wait timeout intervals

- **defaulttimeouts**  
  Return default job, manual feed, and wait timeout intervals

- **setjobtimeout**  
  Set job timeout interval

- **jobtimeout** int  
  Return job timeout interval

- **waittimeout** int  
  Value of wait timeout interval

Device Control

- **pixelsperinch**  
  Set pixel resolution

- **resolution** pixelsperinch  
  Return horizontal pixel resolution

- **processcolors** int  
  Return number of color components

- **setmargins**  
  Set page margins

- **margins** top left  
  Return page margins

- **setmirrorprint**  
  Set mirror printing flag

- **mirrorprint** bool  
  Return mirror printing flag

- **setpagestackorder**  
  Set page stacking order to complement

- **pagestackorder** bool  
  Return complement of page stacking order

- **manualfeed** bool  
  Manual feed flag

- **manualfeedtimeout** int  
  Manual feed timeout interval

Media Selection

- **letter**  
  Select letter-size paper

- **lettersmall**  
  Select letter-size paper with smaller printable area

- **lettertray**  
  Select letter-size paper tray

- **legal**  
  Select legal-size paper

- **legaltray**  
  Select legal-size paper tray

- **ledger**  
  Select ledger-size paper

- **ledgertray**  
  Select ledger-size paper tray
Compatibility

CHAPTER 6

Select 11-by-17-inch paper
Select 11-by-17-inch paper tray
Select A3 paper
Select A3 paper tray
Select A4 paper
Select A4 paper tray
Select A4 paper with smaller printable area
Select A4 paper tray
Select B5 paper
Select B5 paper tray
Select current paper size with smaller printable area

Page Duplexing

bool setduplexmode – Set duplex printing flag
bool duplexmode bool Return duplex printing flag
bool firstside bool Test whether current page prints on front side of output medium
bool newsheet – Print current page if back side
bool settumble – Set tumble flag
bool tumble bool Return tumble flag

Imagesetter Compatibility

bool setaccuratescreens – Set accurate screens flag
bool accuratescreens bool Return accurate screens flag
freq angle checkscreen actualfreq actualangle moirélength
  Return actual screen frequency and angle
width height orientation setpage – Set page dimensions and orientation
left setpagemargin – Set page offset
left pagemargin
width height offset orientation setpageparams – Set page dimensions, offset, and orientation
  width height offset orientation
  Return page dimensions, offset, and orientation
Storage Device Compatibility

- **string devmount bool**
  - Mount storage device
- **string devdismount**
  - Dismount storage device
- **string blocks action devformat**
  - Initialize storage device
- **string devstatus searchable writeable hasNames mounted removable searchOrder free logicalSize true or false**
  - Return storage device attributes
- **proc scratch devforall**
  - Enumerate storage devices

Disk Compatibility

- **diskonline bool**
  - Test whether writeable disk mounted
- **diskstatus free total**
  - Return available disk space
- **blocks action initializedisk**
  - Initialize all writeable disks
- **int setuserdiskpercent**
  - Formerly set percentage of disk reserved for user files
- **userdiskpercent int**
  - Formerly returned percentage of disk reserved for user files

Channel Compatibility

- **int sethardwareiomode**
  - Open channel for communication
- **hardwareiomode int**
  - Return currently enabled communication channel
- **int setsoftwareiomode**
  - Set interpreter and protocol
- **softwareiomode int**
  - Return interpreter and protocol
- **stream emulator emulate**
  - Yield control to emulator
- **stream dict emulator emulate**
  - Yield control to emulator with parameters
- **appletalktype string**
  - Value of AppleTalk type string
- **channel sccbatch baud options**
  - Return serial communication parameters
- **channel sccinteractive baud options**
  - Formerly returned serial communication parameters
- **channel baud options setsccbatch**
  - Set serial communication parameters
- **channel baud options setsccticative**
  - Formerly set serial communication parameters
CHAPTER 6

6.2 Compatibility Operator Details

\texttt{a3} \texttt{-- a3 --} \hspace{1cm} (\texttt{userdict} dictionary)

requests A3-size paper as the output medium. It invokes the \texttt{setpagedevice} operator with a \texttt{PageSize} value of 297 by 420 millimeters ([842 1191] in default user space units) and an \texttt{ImagingBBox} value of \texttt{null}, denoting the largest bounding box possible for this page size on the current output device (see PLR3, p. 414).

This operator sets the \texttt{PageSize} entry in the \texttt{Policies} dictionary to 7, disabling the normal LanguageLevel 2 media selection mechanism and unconditionally establishing A3 as the page size whether or not an A3 media source is available on the device (see PLR3, pp. 434–435).

\textit{Note:} Because this operator invokes \texttt{setpagedevice}, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: \texttt{limitcheck}

See also: \texttt{a3tray}

\texttt{a3tray} \texttt{-- a3tray --} \hspace{1cm} (\texttt{statusdict} dictionary)

requests a paper tray containing A3-size paper as the media source. It invokes the \texttt{setpagedevice} operator with a \texttt{PageSize} value of 297 by 420 millimeters ([842 1191] in default user space units) and an \texttt{ImagingBBox} value of \texttt{null}, denoting the largest bounding box possible for this page size on the current output device (see PLR3, p. 414).

This operator sets the \texttt{PageSize} entry in the \texttt{Policies} dictionary to 0, causing a \texttt{configurationerror} to be raised if an A3 media source is not available on the device (see PLR3, p. 434). For compatibility with LanguageLevel 1, the implementation of the \texttt{a3tray} operator converts any such \texttt{configurationerror} to a \texttt{rangecheck} error.

\textit{Note:} Because this operator invokes \texttt{setpagedevice}, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: \texttt{limitcheck}, \texttt{rangecheck}

See also: \texttt{a3}
requests A4-size paper as the output medium. It invokes the \texttt{setpagedevice} operator with a \texttt{PageSize} value of 210 by 297 millimeters ([595 842] in default user space units) and an \texttt{ImagingBBox} value of \texttt{null}, denoting the largest bounding box possible for this page size on the current output device (see PLR3, p. 414).

This operator sets the \texttt{PageSize} entry in the \texttt{Policies} dictionary to 7, disabling the normal LanguageLevel 2 media selection mechanism and unconditionally establishing A4 as the page size whether or not an A4 media source is available on the device (see PLR3, pp. 434–435).

\textbf{Note:} Because this operator invokes \texttt{setpagedevice}, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: \texttt{limitcheck}
See also: \texttt{a4small}, \texttt{a4tray}

\texttt{a4small} \hspace{1cm} \texttt{a4small} \hspace{1cm} (\texttt{userdict} dictionary)

requests A4-size paper as the output medium, with a margin of approximately 1/3 inch at all four edges of the physical medium. It invokes the \texttt{setpagedevice} operator with a \texttt{PageSize} value of 210 by 297 millimeters ([595 842] in default user space units) and an \texttt{ImagingBBox} value of [25 25 570 817] (inset 25 units from each edge of the physical sheet).

This operator sets the \texttt{PageSize} entry in the \texttt{Policies} dictionary to 7, disabling the normal LanguageLevel 2 media selection mechanism and unconditionally establishing A4 as the page size whether or not an A4 media source is available on the device (see PLR3, pp. 434–435).

\textbf{Note:} Because this operator invokes \texttt{setpagedevice}, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: \texttt{limitcheck}
See also: \texttt{a4small}, \texttt{a4tray}
CHAPTER 6

94 Compatibility

a4tray – a4tray – (statusdict dictionary)

requests a paper tray containing A4-size paper as the media source. It invokes the setpagedevice operator with a PageSize value of 210 by 297 millimeters ([595 842] in default user space units) and an ImagingBBox value of null, denoting the largest bounding box possible for this page size on the current output device (see PLR3, p. 414).

This operator sets the PageSize entry in the Policies dictionary to 0, causing a configurationerror to be raised if an A4 media source is not available on the device (see PLR3, p. 434). For compatibility with LanguageLevel 1, the implementation of the a4tray operator converts any such configurationerror to a rangecheck error.

Note: Because this operator invokes setpagedevice, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: limitcheck, rangecheck
See also: a4, a4small

accuratescreens – accuratescreens bool (statusdict dictionary)

returns the current value of the user parameter AccurateScreens (PLR3, pp. 749 and 757). If this parameter is not present, an undefined error will occur.

Errors: stackoverflow, undefined
See also: setaccuratescreens

appletalktype – appletalktype string (statusdict dictionary)

is a string with the same value as the LocalTalkType device parameter in the %LocalTalk% parameter set and the EtherTalkType device parameter in the %EtherTalk% parameter set (see Tables 3.12 on page 42 and 3.13 on page 43).

appletalktype is not an operator; it is a name in statusdict associated with the AppleTalk type string. LanguageLevel 1 applications that formerly specified the AppleTalk type by setting the value of this string, using code such as

/appletalktype (SpoolerControlledPrinter) def

can continue to do so; the PostScript interpreter will automatically update the values of both the LocalTalkType and EtherTalkType device parameters to match.
Similarly, changes in the value of either the `LocalTalkType` or `EtherTalkType` device parameter will affect the value of the `appletalktype` string.

**Errors:** stackoverflow, undefined

### b5

```plaintext
b5 – b5 – (userdict dictionary)
```

requests B5-size paper as the output medium. It invokes the `setpagedevice` operator with a `PageSize` value representing this size and an `ImagingBBox` value of `null`, denoting the largest bounding box possible for the specified page size on the current output device (see PLR3, p. 414).

**Note:** Two different definitions are in common use for the B5 paper size (see Table 3.27 on page 56). Most PostScript implementations will request a `PageSize` value of `[499 709]` in default user space units (72nds of an inch), corresponding to the International Organization for Standardization (ISO) definition of \(176 \times 250\) mm; implementations intended for the Japanese market will typically set `PageSize` to `[516 729]`, representing the Japanese Industrial Standards Committee (JIS) definition of \(182 \times 257\) mm.

This operator sets the `PageSize` entry in the `Policies` dictionary to 7, disabling the normal LanguageLevel 2 media selection mechanism and unconditionally establishing B5 as the page size whether or not a B5 media source is available on the device (see PLR3, pp. 434–435).

**Note:** Because this operator invokes `setpagedevice`, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

**Errors:** limitcheck

See also: `b5tray`

### b5tray

```plaintext
b5tray – b5tray – (statusdict dictionary)
```

requests a paper tray containing B5-size paper as the media source. It invokes the `setpagedevice` operator with a `PageSize` value representing this size and an `ImagingBBox` value of `null`, denoting the largest bounding box possible for the specified page size on the current output device (see PLR3, p. 414).

**Note:** Two different definitions are in common use for the B5 paper size (see Table 3.27 on page 56). Most PostScript implementations will request a `PageSize` value of `[499 709]` in default user space units (72nds of an inch), corresponding to the
International Organization for Standardization (ISO) definition of \(176 \times 250 \text{ mm}\); implementations intended for the Japanese market will typically set `PageSize` to \([516 \ 729]\), representing the Japanese Industrial Standards Committee (JIS) definition of \(182 \times 257 \text{ mm}\).

This operator sets the `PageSize` entry in the `Policies` dictionary to 0, causing a `configurationerror` to be raised if a B5 media source is not available on the device (see `PLR3`, p. 434). For compatibility with LanguageLevel 1, the implementation of the `b5tray` operator converts any such `configurationerror` to a `rangecheck` error.

**Note:** Because this operator invokes `setpagedevice`, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see `PLR3`, pp. 679–680 and Section 6.2.6 (pp. 426–432).

**Errors:** `limitcheck`, `rangecheck`

See also: `b5`

---

`buildtime` – `buildtime int` (**`statusdict` dictionary**)

is a read-only integer with the same value as the system parameter `BuildTime` (`PLR3`, p. 751). `buildtime` is not an operator; it is a name in `statusdict` associated with the integer value.

**Errors:** `stackoverflow`

---

`byteorder` – `byteorder bool` (**`statusdict` dictionary**)

is a read-only boolean value with the same value as the system parameter `ByteOrder` (`PLR3`, p. 751). `byteorder` is not an operator; it is a name in `statusdict` associated with the boolean value.

**Errors:** `stackoverflow`

---

`checkpassword` `string checkpassword bool` `int checkpassword bool` (**`statusdict` dictionary**)

checks whether the operand is a valid password for the `startjob` or `exitserver` operator. `checkpassword` returns `true` if the operand is equal to either the `System-ParamsPassword` or `StartJobPassword` system parameter; see `PLR3`, Section C.3.1
(pp. 754–755). (If an integer operand is supplied, it is converted to a string for purposes of password comparison.) If either SystemParamsPassword or StartJobPassword is not set (that is, is the empty string), checkpassword returns true.

There is no LanguageLevel 2 equivalent for this operator.

Errors: stackunderflow, typecheck

checkscreen freq angle checkscreen actualfreq actualangle moirélength (statusdict dictionary)

returns the actual screen frequency and angle that would result if the setscreen operator were called with the operand values freq and angle. The values returned for actualfreq and actualangle are the same as for the ActualFrequency and ActualAngle entries in a type 1 halftone dictionary (PLR3, pp. 487–488). moirélength is the distance in inches at which the deviation from the requested dot pattern would reach a fixed fraction of the cell size, and is thus a measure of how accurately the actual screen would approximate the requested screen. Note that this operator does not affect the current screen.

There is no LanguageLevel 2 equivalent for this operator.

Errors: stackoverflow, stackunderflow, typecheck

defaulttimeouts – defaulttimeouts job manualfeed wait (statusdict dictionary)

returns the current values of the system parameters JobTimeout and WaitTimeout (see Table 3.2 on page 15) and the page device parameter ManualFeedTimeout (Table 2.2 on page 5) as job, wait, and manualfeed, respectively. This operator always succeeds and always returns three result values; if any of the required parameters is not present, the corresponding result value is 0.

Errors: stackoverflow
See also: setdefaulttimeouts

devdismount string devdismount –

attempts to dismount the storage device identified by string by setting the device parameter Mounted (see Table 3.32 on page 61) to false in the parameter set corresponding to the device. If the device is not currently mounted, this operator has
Compatibility

CHAPTER 6

98

Compatibility

no effect. Some devices cannot be dismounted; attempting to dismount such a de-
vice will also have no effect.

This operator should be invoked only from within a system administrator job.

Errors: invalidaccess, stackunderflow, typecheck, undefinedfilename

\texttt{devforall} \quad \texttt{proc scratch devforall} –

enumerates all known storage devices. For each device, \texttt{devforall} copies the de-
vice's name into the supplied \texttt{scratch} string, pushes a string object designating the
substring of \texttt{scratch} actually used, and calls \texttt{proc}. \texttt{devforall} does not return any re-
results of its own, but \texttt{proc} may do so.

\textbf{Note:} Some of the storage devices enumerated by \texttt{devforall} correspond to communication channels. The value of the \texttt{HasNames} parameter for all such devices will be false (see Table 3.3 on page 29).

There is no LanguageLevel 2 equivalent for this operator (although its effect can be approximated by applying \texttt{resourceforall} to the \texttt{IODevice} resource category).

Errors: invalidaccess, rangecheck, stackoverflow, stackunderflow, typecheck, undefined

\texttt{devformat} \quad \texttt{string blocks action devformat} –

initializes the storage device identified by \texttt{string} by setting the device parameters
\texttt{LogicalSize} and \texttt{InitializeAction} (see Tables 3.32–3.35 on pages 61–66) to the
values \texttt{blocks} and \texttt{action} + 1, respectively, in the parameter set corresponding to
the device.

This operator should be invoked only from within a system administrator job.

Errors: invalidaccess, limitcheck, rangecheck, stackunderflow, typecheck, undefined, undefinedfilename

\texttt{devmount} \quad \texttt{string devmount bool}

attempts to mount the storage device identified by \texttt{string} by setting the device parameter \texttt{Mounted} (see Table 3.32 on page 61) to \texttt{true} in the parameter set corresponding to the device. The boolean result \texttt{bool} is set to \texttt{true} if the device is
successfully mounted (or was already mounted), or to false if the device cannot be mounted at this time.

This operator should be invoked only from within a system administrator job.

Errors: invalidaccess, stackunderflow, undefinedfilename

devstatus string devstatus searchable writeable hasNames mounted
  removable searchOrder free logicalSize true (if device known)
  false (if not)

returns the values of various file system attributes for the disk device identified by string (which must be %diskn% for some integer value of n). If the specified device name is known, the boolean value true is returned on the top of the stack, accompanied by the values of the device parameters Searchable, Writeable, HasNames, Mounted, Removable, SearchOrder, Free, and LogicalSize (see Table 3.32 on page 61). If the device name is unknown, only the value false is returned on the stack.

Errors: stackoverflow, stackunderflow, typecheck

diskonline – diskonline bool (statusdict dictionary)

returns a boolean value indicating whether there exists a writeable disk device. The result will be true if there exists a device parameter set named %diskn% (for some integer value of n) in which the value of the Writeable parameter is true; if no such parameter set exists, a false value will be returned. Note that the writeable disk device found need not have an initialized file system.

Errors: stackoverflow

diskstatus – diskstatus free total (statusdict dictionary)

returns the current number of free blocks and the total number of blocks available on all writeable disk devices. These values are determined by finding all known device parameter sets named %diskn% (for integer values of n) in which the value of the Writeable parameter is true, and totaling the values of their Free and LogicalSize parameters, respectively (see Table 3.32 on page 61).

Errors: stackoverflow
doprinterrors – doprinterrors bool

returns the current value of the system parameter DoPrintErrors (see Table 3.2 on page 15). If this parameter is not present, an undefined error will occur.

Errors: stackoverflow, undefined
See also: setdoprinterrors

dostartpage – dostartpage bool

returns the current value of the system parameter DoStartPage (see Table 3.2 on page 15). If this parameter is not present, an undefined error will occur.

Errors: stackoverflow, undefined
See also: setdostartpage

dosysstart – dosysstart bool

returns a boolean value indicating whether a startup action is to be performed during subsequent restarts of the interpreter. The result will be true if the value of the system parameter StartupMode (PLR3, p. 753) is nonzero and false if it is zero. If this parameter is not present, an undefined error will occur.

Errors: stackoverflow, undefined
See also: setdosysstart

duplexmode – duplexmode bool

returns the current value of the page device parameter Duplex (PLR3, p. 416). If the Duplex page device parameter is not present, an undefined error will occur.

Errors: stackoverflow, undefined
See also: setduplexmode
emulate stream emulator emulate – (statusdict dictionary)
stream dict emulator emulate –

causes the PostScript interpreter to yield control to the emulator named by `emulator`. The allowed values of `emulator` are defined by the implicit resource category `Emulator` (see Section 4.2, “Implicit Resource Categories.”). An illegal emulation name will cause a `rangecheck` error.

This operator is present only on output devices in which one or more emulators are coresident with the PostScript interpreter. The exact details of such emulators are product-dependent and may vary on different devices, even though the emulation name may be the same. In most coresident emulators, the command sequence `Esc Del 0` can be used to return control to the PostScript interpreter; however, the PostScript language context will generally have been lost.

The `stream` operand is a file object that provides the input source for the emulator. The contents of the input stream must be appropriate to the emulator, or an `invalidaccess` error will occur.

The optional `dict` operand is a dictionary holding any parameters the specified emulator may need. If the emulator requires parameters and `dict` is omitted, product-dependent defaults will be used if possible. If the emulator does not require parameters, `dict` will be ignored if present. Currently, no emulators require parameters.

There is no LanguageLevel 2 equivalent for this operator.

Errors: invalidaccess, rangecheck, stackoverflow, stackunderflow

firstside – firstside bool (statusdict dictionary)

returns a boolean value indicating whether the current page is to be printed on the front side (`true`) or the back side (`false`) of the output medium.

Note: This operator is sometimes found on devices that do not support duplex printing. On such devices, it may be used in generating output that is intended to be copied using a duplex copier.

There is no LanguageLevel 2 equivalent for this operator.

Errors: stackoverflow
**hardwareiomode** – hardwareiomode int *(statusdict dictionary)*

returns an integer code identifying the currently enabled communication channel (that is, the channel for which the value of the Enabled parameter in the corresponding device parameter set is true). The possible codes are as follows:

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

This operator will always return the channel identified by the system parameter CurInputDevice if that channel is on and enabled and is one of the ones listed above. Otherwise, it will return the smallest code in the list denoting a channel that is on and enabled. If no such channel exists, it will return 0.

**Errors:** stackoverflow

See also: sethardwareiomode

**initializedisk** blocks action initializedisk – *(statusdict dictionary)*

initializes each writeable disk by setting the device parameters LogicalSize and InitializeAction in each %diskn% parameter set (see Table 3.32 on page 61) to the values blocks and action + 1, respectively.

This operator should be invoked only from within a system administrator job.

**Errors:** invalidaccess, ioerror, rangecheck, stackunderflow, typecheck

**jobname** – jobname string *(statusdict dictionary)*

is a string with the same value as the user parameter JobName (PLR3, p. 750).

**jobname** is not an operator; it is a name in statusdict associated with the job name string. LanguageLevel 1 applications that formerly specified the job name by setting the value of this string, using code such as

`/jobname (Job 1) def`

can continue to do so; the PostScript interpreter will automatically update the value of the JobName user parameter to match. Similarly, changes in the value of the JobName user parameter will affect the value of the jobname string.

**Errors:** stackoverflow, undefined
jobtimeout – jobtimeout int

returns the current value of the user parameter JobTimeout (see Table 3.1 on page 14). If this parameter is not present, an undefined error will occur.

Errors: stackoverflow, undefined
See also: setjobtimeout

ledger – ledger –

requests ledger-size paper as the output medium. It invokes the setpagedevice operator with a PageSize value of 17 by 11 inches ([1224 792] in default user space units) and an ImagingBBox value of null, denoting the largest bounding box possible for this page size on the current output device (see PLR3, p. 414).

This operator sets the PageSize entry in the Policies dictionary to 7, disabling the normal LanguageLevel 2 media selection mechanism and unconditionally establishing ledger size as the page size whether or not a ledger-size media source is available on the device (see PLR3, pp. 434–435).

Note: Because this operator invokes setpagedevice, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: limitcheck
See also: ledgertray

ledgertray – ledgertray –

requests a paper tray containing ledger-size paper as the media source. It invokes the setpagedevice operator with a PageSize value of 17 by 11 inches ([1224 792] in default user space units) and an ImagingBBox value of null, denoting the largest bounding box possible for this page size on the current output device (see PLR3, p. 414).

This operator sets the PageSize entry in the Policies dictionary to 0, causing a configurationerror to be raised if a ledger-size media source is not available on the device (see PLR3, p. 434). For compatibility with LanguageLevel 1, the implementation of the ledgertray operator converts any such configurationerror to a rangecheck error.
Note: Because this operator invokes `setpagedevice`, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: limitcheck, rangecheck
See also: ledger

```
legal – legal –  (userdict dictionary)
```

requests legal-size paper as the output medium. It invokes the `setpagedevice` operator with a `PageSize` value of 8.5 by 14 inches ([612 1008] in default user space units) and an `ImagingBBox` value of `null`, denoting the largest bounding box possible for this page size on the current output device (see PLR3, p. 414).

This operator sets the `PageSize` entry in the `Policies` dictionary to 7, disabling the normal LanguageLevel 2 media selection mechanism and unconditionally establishing legal size as the page size whether or not a legal-size media source is available on the device (see PLR3, pp. 434–435).

Note: Because this operator invokes `setpagedevice`, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: limitcheck
See also: legaltray

```
legaltray – legaltray –  (statusdict dictionary)
```

requests a paper tray containing legal-size paper as the media source. It invokes the `setpagedevice` operator with a `PageSize` value of 8.5 by 14 inches ([612 1008] in default user space units) and an `ImagingBBox` value of `null`, denoting the largest bounding box possible for this page size on the current output device (see PLR3, p. 414).

This operator sets the `PageSize` entry in the `Policies` dictionary to 0, causing a `configurationerror` to be raised if a legal-size media source is not available on the device (see PLR3, p. 434). For compatibility with LanguageLevel 1, the implementation of the `legaltray` operator converts any such `configurationerror` to a `rangecheck` error.
Note: Because this operator invokes `setpagedevice`, it also causes all of that operator's attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: limitcheck, rangecheck
See also: legal

letter – letter – (userdict dictionary)

requests letter-size paper as the output medium. It invokes the `setpagedevice` operator with a `PageSize` value of 8.5 by 11 inches ([612 792] in default user space units) and an `ImagingBBox` value of `null`, denoting the largest bounding box possible for this page size on the current output device (see PLR3, p. 414).

This operator sets the `PageSize` entry in the `Policies` dictionary to 7, disabling the normal LanguageLevel 2 media selection mechanism and unconditionally establishing letter size as the page size whether or not a letter-size media source is available on the device (see PLR3, pp. 434–435).

Note: Because this operator invokes `setpagedevice`, it also causes all of that operator's attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: limitcheck
See also: lettersmall, lettertray

lettersmall – lettersmall – (userdict dictionary)

requests letter-size paper as the output medium, with a margin of approximately 1/3 inch at all four edges of the physical medium. It invokes the `setpagedevice` operator with a `PageSize` value of 8.5 by 11 inches ([612 792] in default user space units) and an `ImagingBBox` value of [25 25 587 767] (inset 25 units from each edge of the physical sheet).

This operator sets the `PageSize` entry in the `Policies` dictionary to 7, disabling the normal LanguageLevel 2 media selection mechanism and unconditionally establishing letter size as the page size whether or not a letter-size media source is available on the device (see PLR3, pp. 434–435).

Note: Because this operator invokes `setpagedevice`, it also causes all of that operator's attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).
Errors: limitcheck
See also: letter, lettertray

lettertray – lettertray – (statusdict dictionary)

requests a paper tray containing letter-size paper as the media source. It invokes
the setpagedevice operator with a PageSize value of 8.5 by 11 inches ([612 792]
in default user space units) and an ImagingBBox value of null, denoting the largest
bounding box possible for this page size on the current output device (see PLR3,
p. 414).

This operator sets the PageSize entry in the Policies dictionary to 0, causing a
configurationerror to be raised if a letter-size media source is not available on the
device (see PLR3, p. 434). For compatibility with LanguageLevel 1, the implementa-
tion of the lettertray operator converts any such configurationerror to a
rangecheck error.

Note: Because this operator invokes setpagedevice, it also causes all of that operator’s
attendant side effects, such as reinitializing the graphics state and erasing the current
page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: limitcheck, rangecheck
See also: letter, lettersmall

manualfeed – manualfeed bool (statusdict dictionary)

is a boolean value that works in conjunction with the page device parameter
ManualFeed (PLR3, pp. 402–403) to determine whether a sheet of input medium
is to be fed manually. If the value of either manualfeed or ManualFeed is true at
the time of a showpage or copypage operation, then that page will be fed manu-
ally; otherwise, the page will be fed automatically.

manualfeed is not an operator; it is a name in statusdict associated with the bool-
ean value. The values of manualfeed and ManualFeed are independent; setting ei-
ther value does not affect the other. The initial value of manualfeed at power-on is
false.

Errors: stackoverflow, undefined
**manualfeedtimeout** – **manualfeedtimeout int**  
(*statusdict dictionary*)

is an integer that works in conjunction with the page device parameter **ManualFeedTimeout** (see Table 2.2 on page 5) to determine the manual feed timeout interval.

**manualfeedtimeout** is not an operator; it is a name in **statusdict** associated with the integer timeout value. The values of **manualfeedtimeout** and **ManualFeedTimeout** are independent; setting either value does not affect the other.

By default, **manualfeedtimeout** is not defined and the value of the page device parameter **ManualFeedTimeout** is used to determine the timeout interval. If a job has defined a **manualfeedtimeout** entry in **statusdict**, the value of this entry will be used for the timeout interval in place of the **ManualFeedTimeout** parameter.

**Errors:** stackoverflow

**margins** – **margins top left**  
(*statusdict dictionary*)

returns the values of the top and left page margins, taken from the page device parameter **Margins** (PLR3, p. 415). If this parameter is not present, an undefined error will occur.

**Errors:** stackoverflow, undefined

**See also:** setmargins

**mirrorprint** – **mirrorprint bool**  
(*statusdict dictionary*)

returns the current value of the page device parameter **MirrorPrint** (PLR3, p. 415). If this parameter is not present, an undefined error will occur.

**Errors:** stackoverflow, undefined

**See also:** setmirrorprint

**newsheet** – **newsheet** –  
(*statusdict dictionary*)

causes the current page to be printed if it is a back side and the page device parameter **Duplex** (PLR3, p. 416) is true; otherwise has no effect. The page is printed as is (perhaps blank). The operator then sets up a clean printing environment for the
next page. If the **Duplex** page device parameter is not present, an **undefined** error will occur.

There is no LanguageLevel 2 equivalent for this operator (although its effect can be approximated by invoking **setpagedevice** with an empty request dictionary).

**Errors:** undefined  
**See also:** firstside

---

**note**  
(**userdict** dictionary)

requests the same paper size already established as the current output medium, but with a margin of approximately 1/3 inch at all four edges of the physical medium. This operator invokes **setpagedevice** with an **ImagingBBox** value of \[25\ 25\ (width – 25)\ (height – 25)\] (inset 25 units from each edge of the physical sheet), where the dimensions *width* and *height* are taken from the current value of the page device parameter **PageSize**. If the current page size is letter or A4, the **note** operator is equivalent to **lettersmall** or **a4small**, respectively.

This operator sets the **PageSize** entry in the **Policies** dictionary to 7, disabling the normal LanguageLevel 2 media selection mechanism and unconditionally establishing the specified page size, regardless of whether a media source of that size is available on the device (see **PLR3**, pp. 434–435).

**Note:** Because this operator invokes **setpagedevice**, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see **PLR3**, pp. 679–680 and Section 6.2.6 (pp. 426–432).

**Errors:** limitcheck

---

**pagecount**  
(**statusdict** dictionary)

returns the current value of the system parameter **PageCount** (**PLR3**, p. 752). If this parameter is not present, an **undefined** error will occur.

**Errors:** stackoverflow, undefined
pagemargin – pagemargin left

returns the horizontal \((x)\) component of the page device parameter \(PageOffset\) (PLR3, p. 415), measured in default user space units. If this parameter is not present, an undefined error will occur.

Errors: stackoverflow, undefined
See also: setpagemargin

pageparams – pageparams width height offset orientation

returns information about the dimensions of the physical medium, the page offset, and the page image orientation that were assumed during the generation of the current page description. The return values \(width\), \(height\), and \(offset\) are taken from the page device parameters \(PageSize\) and \(PageOffset\) (PLR3, pp. 401 and 415, respectively). However, the value returned for \(orientation\) is determined by device-dependent means and is not based on the value of the Orientation page device parameter.

If the \(orientation\) value returned is 0 (normal default orientation), \(width\) and \(height\) return the vertical \((y)\) and horizontal \((x)\) components, respectively, of \(PageSize\). If \(orientation\) is 1 (rotated 90 degrees counterclockwise), the dimensions are exchanged, with \(width\) equal to the \(x\) component of \(PageSize\) and \(height\) equal to the \(y\) component. (Note that the roles of the \(width\) and \(height\) operands are the opposite of what the page orientation would seem to imply.) Orientation values of 2 (rotated 180 degrees) or 3 (rotated 270 degrees counterclockwise, equivalent to 90 degrees clockwise), as described in PLR3, p. 412, are never returned as the value of \(orientation\); pages in these orientations are reported as if the Orientation value were 0 or 1 instead.

The value returned for \(offset\) is either the horizontal \((x)\) or vertical \((y)\) component of the \(PageOffset\) page device parameter, depending on the orientation. The correspondence between the value of \(orientation\) and the component of \(PageOffset\) returned is device-dependent and may vary from one output device to another.

Errors: stackoverflow, undefined
See also: setpageparams, setpage
pagestackorder \( \text{bool} \) \( \text{statusdict} \) \( \text{dictionary} \)

returns the logical complement of the page device parameter \text{OutputFaceUp} \ (PLR3, p. 418)—that is, the value returned for \text{bool} \ is \ text{true} \ if \text{OutputFaceUp} \ is \ text{false} \ and \text{false} \ if \text{OutputFaceUp} \ is \text{true}. If the \text{OutputFaceUp} \ parameter is not present, an \text{undefined} \ error will occur.

Errors: stackoverflow, undefined
See also: setpagestackorder

printername \( \text{string} \) \( \text{substring} \) \( \text{statusdict} \) \( \text{dictionary} \)

copies the value of the system parameter \text{PrinterName} \ (PLR3, p. 753) into \text{string} \ and returns a string object designating the substring of \text{string} \ actually used. If the \text{PrinterName} \ parameter is not present, an \text{undefined} \ error will occur.

Errors: rangecheck, stackunderflow, typecheck, undefined
See also: setprintername

processcolors \( \text{int} \) \( \text{statusdict} \) \( \text{dictionary} \)

returns the number of color components in the current page device’s process color model (1 for \text{DeviceGray}, 3 for \text{DeviceRGB} or \text{DeviceCMY}, 4 for \text{DeviceCMYK} or \text{DeviceRGBK}, or the number of colorants specified in a \text{DeviceN} \ color model; see PLR3, pp. 422–424).

Note: This operator is mandatory on devices that can produce more than one color; it is optional (and typically is not present) on monochrome devices. Its absence indicates a monochrome device (one process color).

Errors: stackoverflow

product \( \text{string} \) \( \text{statusdict} \) \( \text{dictionary} \)

is a string object representing the name of the product on which the PostScript interpreter is running. \text{product} \ is not an operator; it is a name in \text{statusdict} \ associated with the string object. This string is initialized to the same value returned by the standard \text{systemdict} \ operator \text{product} \ (PLR3, p. 634).

Errors: stackoverflow
6.2 Compatibility Operator Details

**ramsize** – *ramsize int*  
*(statusdict dictionary)*

returns the number of bytes of installed RAM available to the page description language, taken from the RamSize system parameter (see Table 3.2 on page 15). If this parameter is not present, an **undefined** error will occur.

**Errors:** stackoverflow, undefined

**realformat** – *realformat string*  
*(statusdict dictionary)*

is a read-only string with the same value as the system parameter RealFormat (PLR3, p. 753). **realformat** is not an operator; it is a name in **statusdict** associated with the string object.

**Errors:** stackoverflow

**resolution** – *resolution pixelsperinch*  
*(statusdict dictionary)*

returns the horizontal resolution of the current output device in pixels per inch, taken from the horizontal (x) component of the HWResolution page device parameter (PLR3, p. 414). If this parameter is not present, an **undefined** error will occur.

**Errors:** stackoverflow, undefined  
**See also:** setresolution

**revision** – *revision int*  
*(statusdict dictionary)*

is a read-only integer with the same value as the system parameter Revision (PLR3, p. 753). **revision** is not an operator; it is a name in **statusdict** associated with the integer value.

**Errors:** stackoverflow
sccbatch channel sccbatch baud options (statusdict dictionary)

returns the values of device parameters for serial communication. The channel operand identifies the parameter set affected:

<table>
<thead>
<tr>
<th>channel</th>
<th>PARAMETER SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>%SerialB_NV%</td>
</tr>
<tr>
<td>25</td>
<td>%Serial_NV%</td>
</tr>
</tbody>
</table>

baud returns the value of the Baud device parameter; options is a 1-byte integer (0–255) that encodes the values of the StopBits, DataBits, FlowControl, and Parity parameters as equivalent LanguageLevel 1 settings. (See Table 3.4 on page 33 for descriptions of all of these device parameters.) The settings are encoded as follows, where the bit positions within the options byte are numbered from 7 (high-order) to 0 (low-order):

<table>
<thead>
<tr>
<th>BIT 7</th>
<th>STOP BITS</th>
<th>DATA BITS</th>
<th>FLOW CONTROL</th>
<th>PARITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 stop bit</td>
<td>0 Standard</td>
<td>XON/XOFF</td>
<td>Space</td>
</tr>
<tr>
<td>1</td>
<td>2 stop bits</td>
<td>7 bits</td>
<td>DTR</td>
<td>Odd</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 bits</td>
<td>ETX/ACK</td>
<td>Even</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mark</td>
</tr>
</tbody>
</table>

The value of the LanguageLevel 2 StopBits parameter maps directly into the equivalent LanguageLevel 1 settings shown above. For the FlowControl parameter, the values XonXoff, XonXoff2, and RobustXonXoff all map to an options setting of 0 (XON/XOFF), Dtr or DtrLow to 1 (DTR), and EtxAck to 2 (ETX/ACK).

For the DataBits and Parity parameters, the correspondence is less straightforward:

<table>
<thead>
<tr>
<th>LANGUAGELEVEL 2</th>
<th>LANGUAGELEVEL 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataBits</td>
<td>Parity</td>
</tr>
<tr>
<td>7</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Space</td>
</tr>
<tr>
<td></td>
<td>Mark</td>
</tr>
<tr>
<td></td>
<td>Odd</td>
</tr>
<tr>
<td></td>
<td>Even</td>
</tr>
<tr>
<td>8</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Space</td>
</tr>
<tr>
<td></td>
<td>Mark</td>
</tr>
<tr>
<td></td>
<td>Odd</td>
</tr>
<tr>
<td></td>
<td>Even</td>
</tr>
</tbody>
</table>

Note: These mappings are intended to provide the best attainable compatibility with LanguageLevel 1 behavior, but in some cases no exact equivalent is possible. For ex-
ample, LanguageLevel 1 does not support the combination of 7 data bits with no parity (that is, 7 bits total for data and parity together). Thus the combination of DataBits equal to 7 with Parity equal to None is imperfectly mapped to 7 data bits with mark parity.

Errors: rangecheck, stackoverflow, stackunderflow, typecheck
See also: setsccbatch

sccinteractive  channel sccinteractive baud options  (statusdict dictionary)

formerly returned the values of device parameters for serial communication in interactive or emulation mode. This operator now pops the channel operand off the stack, pushes zero values on the stack for baud and options, and otherwise does nothing.

There is no LanguageLevel 2 equivalent for this operator.

Errors: invalidaccess, rangecheck, stackoverflow, stackunderflow, typecheck
See also: setsccinteractive

setaccuratescreens  bool setaccuratescreens –  (statusdict dictionary)

sets the value of the user parameter AccurateScreens (PLR3, pp. 749 and 757) to bool. If this parameter is not present, setaccuratescreens does nothing.

Errors: invalidaccess, stackunderflow, typecheck, undefined
See also: accuratescreens

setdefaulttimeouts  job manualfeed wait setdefaulttimeouts –  (statusdict dictionary)

sets the values of the system parameters JobTimeout and WaitTimeout (see Table 3.2 on page 15), and the page device parameter ManualFeedTimeout (Table 2.2 on page 5) to job, wait, and manualfeed, respectively.

This operator always accepts three operands, even if one or more of the relevant system or page device parameters are not present. If JobTimeout or WaitTimeout is absent, the corresponding operand is ignored; if ManualFeedTimeout is absent, the Policies dictionary is consulted and the applicable recovery policy is invoked (see PLR3, Section 6.2.7).
This operator should be invoked only from within a system administrator job.

Note: Because this operator invokes `setpagedevice`, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: invalidaccess, rangecheck, stackunderflow, typecheck
See also: defaulttimeouts

`setdoprinterrors` \(\text{bool setdoprinterrors} \rightarrow \text{(statusdict dictionary)}\)

sets the value of the system parameter `DoPrintErrors` (see Table 3.2 on page 15) to `bool`. If this parameter is not present, `setdoprinterrors` does nothing.

This operator should be invoked only from within a system administrator job.

Errors: invalidaccess, stackunderflow, typecheck, undefined
See also: doprinterrors

`setdostartpage` \(\text{bool setdostartpage} \rightarrow \text{(statusdict dictionary)}\)

sets the value of the system parameter `DoStartPage` (see Table 3.2 on page 15) to `bool`. If this parameter is not present, `setdostartpage` does nothing.

This operator should be invoked only from within a system administrator job.

Errors: invalidaccess, stackunderflow, typecheck, undefined
See also: dostartpage

`setdosysstart` \(\text{bool setdosysstart} \rightarrow \text{(statusdict dictionary)}\)

sets the value of the system parameter `StartupMode` (PLR3, p. 753) according to the value of `bool`: to 1 if `bool` is `true` and 0 if it is `false`. If this parameter is not present, `setdosysstart` does nothing.

This operator should be invoked only from within a system administrator job.

Errors: invalidaccess, stackunderflow, typecheck, undefined
See also: dosysstart
**setduplexmode**  
```
bool setduplexmode -
```

*(statusdict dictionary)*

sets the value of the page device parameter *Duplex* (PLR3, p. 416) to *bool*. If this parameter is not present, the *Policies* dictionary is consulted and the applicable recovery policy is invoked (see PLR3, Section 6.2.7).

**Note:** Because this operator invokes *setpagedevice*, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

**Errors:** stackunderflow, typecheck, undefined  
**See also:** duplexmode

---

**sethardwareiomode**  
```
int sethardwareiomode -
```

*(statusdict dictionary)*

opens the specified channel(s) for communication. The operand *int* is a code identifying the channels to be opened:

- 0  %Serial% and %SerialB%
- 1  %Parallel%
- 2  %LocalTalk% or %EtherTalk% (or both if they exist)
- 3  %Serial% and %SerialB%

The specified channels are opened by setting their *On* and *Enabled* parameters (see Table 3.3 on page 29) to *true*; all other channels are explicitly closed by setting their *On* and *Enabled* parameters to *false*.

This operator should be invoked only from within a system administrator job.

**Errors:** invalidaccess, rangecheck, stackunderflow, typecheck  
**See also:** hardwareiomode

---

**setjobtimeout**  
```
int setjobtimeout -
```

*(statusdict dictionary)*

sets the value of the user parameter *JobTimeout* (see Table 3.1 on page 14) to *int*. If this parameter is not present, *setjobtimeout* does nothing.

**Errors:** stackunderflow, typecheck, undefined  
**See also:** jobtimeout
CHAPTER 6 Compatibility

setmargins  top left (statusdict dictionary)

sets the value of the page device parameter Margins (PLR3, p. 415) to [left top]. If this parameter is not present, the Policies dictionary is consulted and the applicable recovery policy is invoked (see PLR3, Section 6.2.7).

This operator should be invoked only from within a system administrator job.

Note: Because this operator invokes setpagedevice, it also causes all of that operator's attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: invalidaccess, rangecheck, stackunderflow, typecheck, undefined
See also: margins

setmirrorprint  bool (statusdict dictionary)

sets the value of the page device parameter MirrorPrint (PLR3, p. 415) to bool. If this parameter is not present, the Policies dictionary is consulted and the applicable recovery policy is invoked (see PLR3, Section 6.2.7).

Note: Because this operator invokes setpagedevice, it also causes all of that operator's attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: stackunderflow, typecheck
See also: mirrorprint

setpage  width height orientation (statusdict dictionary)

sets the values of the page device parameters PageSize and Orientation (PLR3, pp. 401 and 412, respectively) as specified by the operands. If either of these parameters is not present, the Policies dictionary is consulted and the applicable recovery policy is invoked (see PLR3, Section 6.2.7).

The value supplied for orientation must be either 0 (normal default orientation) or 1 (rotated 90 degrees clockwise). The page device parameter Orientation is set to a corresponding value appropriate to the particular output device. (The exact correspondence between the value of the orientation operand and that of the Orientation page device parameter is device-dependent and may vary from one output device to another.) Values of orientation corresponding to Orientation settings of 2 (ro-
tated 180 degrees) or 3 (rotated 270 degrees counterclockwise, equivalent to 90 degrees clockwise) are not supported.

The PageSize page device parameter is set to \([height \ width]\) if the value of orientation is 0, or to \([width \ height]\) if orientation is 1. (Note that the roles of the width and height operands are the opposite of what the page orientation would seem to imply.)

**Note:** Because this operator invokes `setpagedevice`, it also causes all of that operator's attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

**Errors:** `limitcheck`, `rangecheck`, `stackunderflow`, `typecheck`

**See also:** `pageparams`, `setpageparams`

### setpagemargin

**Syntax:**

```latex
left \text{setpagemargin} \rightarrow \text{(statusdict dictionary)}
```

**Description:** Sets the value of the page device parameter PageOffset (PLR3, p. 415) to \([left \ 0]\). If this parameter is not present, the Policies dictionary is consulted and the applicable recovery policy is invoked (see PLR3, Section 6.2.7).

**Note:** Because this operator invokes `setpagedevice`, it also causes all of that operator's attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

**Errors:** `rangecheck`, `stackunderflow`, `typecheck`

**See also:** `pagemargin`

### setpageparams

**Syntax:**

```latex
width \ height \ offset \ orientation \ \text{setpageparams} \rightarrow \text{(statusdict dictionary)}
```

**Description:** Sets the values of the page device parameters PageSize, PageOffset, and Orientation (PLR3, pp. 401, 415, and 412, respectively) as specified by the operands. If any of these parameters is not present, the Policies dictionary is consulted and the applicable recovery policy is invoked (see PLR3, Section 6.2.7).

The value supplied for orientation must be either 0 (normal default orientation) or 1 (rotated 90 degrees clockwise). The page device parameter Orientation is set to a corresponding value appropriate to the particular output device. (The exact correspondence between the value of the orientation operand and that of the Orientation page device parameter is device-dependent and may vary from one output device to another.) Values of orientation corresponding to Orientation settings of 2 (ro-
tated 180 degrees) or 3 (rotated 270 degrees counterclockwise, equivalent to 90 degrees clockwise) are not supported.

The PageSize page device parameter is set to [height width] if the value of orientation is 0, or to [width height] if orientation is 1. (Note that the roles of the width and height operands are the opposite of what the page orientation would seem to imply.) Either the horizontal (x) or the vertical (y) component of the PageOffset page device parameter is set to the value of the offset operand; the choice of which component to set is device-dependent.

Note: Because this operator invokes setpagedevice, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: limitcheck, rangecheck, stackunderflow, typecheck, undefinedresult
See also: pageparams, setpage

setpagestackorder bool setpagestackorder – (statusdict dictionary)

sets the value of the page device parameter OutputFaceUp (PLR3, p. 418) to the logical complement of bool—that is, to true if bool is false and to false if bool is true. If the OutputFaceUp page device parameter is not present, the Policies dictionary is consulted and the applicable recovery policy is invoked (see PLR3, Section 6.2.7).

This operator should be invoked only from within a system administrator job.

Note: Because this operator invokes setpagedevice, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: invalidaccess, stackunderflow, typecheck, undefined
See also: pagestackorder

setprintername string setprintername – (statusdict dictionary)

sets the value of the system parameter PrinterName (PLR3, p. 753) to string. If this parameter is not present, setprintername does nothing.

This operator should be invoked only from within a system administrator job.

Errors: invalidaccess, limitcheck, stackunderflow, typecheck, undefined
See also: printername
setresolution \textit{pixelsperinch setresolution} – \textit{(statusdict dictionary)}

sets the value of the page device parameter \texttt{HWResolution} (PLR3, p. 414) to \texttt{[pixelsperinch pixelsperinch]}. If this parameter is not present, the Policies dictionary is consulted and the applicable recovery policy is invoked (see PLR3, Section 6.2.7).

\textbf{Note:} Because this operator invokes \texttt{setpagedevice}, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

\textbf{Errors:} rangecheck, stackunderflow, typecheck

See also: resolution

\textit{setsccbatch} \textit{channel baud options setsccbatch} – \textit{(statusdict dictionary)}

sets the values of device parameters for serial communication. The channel operand identifies the parameter set affected:

\begin{center}
\begin{tabular}{|c|c|}
\hline
\textbf{channel} & \textbf{PARAMETER SET} \\
\hline
9 & %SerialB_NV% \\
25 & %Serial_NV% \\
\hline
\end{tabular}
\end{center}

The device parameters affected are \texttt{Baud}, \texttt{StopBits}, \texttt{DataBits}, \texttt{FlowControl}, \texttt{Parity}, and \texttt{CheckParity} (see Table 3.4 on page 33).

\texttt{baud} specifies the value of the \texttt{Baud} device parameter. \texttt{options} is a 1-byte integer (0–255) that encodes the LanguageLevel 1 settings for stop bits, data bits, flow control, and parity. The settings are specified as follows, where the bit positions within the \texttt{options} byte are numbered from 7 (high-order) to 0 (low-order):

\begin{center}
\begin{tabular}{|c|c|c|c|c|c|}
\hline
\textbf{BIT} & \textbf{STOP BITS} & \textbf{DATA BITS} & \textbf{FLOW CONTROL} & \textbf{PARITY} \\
7 & 0 & 0 & 0 & Space \\
6–5 & 1 stop bit & Standard & XON/XOFF & 0 \\
4–3 & 2 stop bits & 1 & DTR & Odd \\
1–0 & 8 bits & 2 & ETX/ACK & Even \\
0 & 2 & 3 & Mark \\
\hline
\end{tabular}
\end{center}

The LanguageLevel 1 settings shown for stop bits and flow control map directly to corresponding LanguageLevel 2 values of the \texttt{StopBits} and \texttt{FlowControl} device parameters, but the mapping for data bits and parity is less straightforward:
**COMPATIBILITY**

**CHAPTER 6**

<table>
<thead>
<tr>
<th>LANGUAGELEVEL 1</th>
<th>LANGUAGELEVEL 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DATA BITS</strong></td>
<td><strong>Parity</strong></td>
</tr>
<tr>
<td>Standard</td>
<td>Space</td>
</tr>
<tr>
<td></td>
<td>Mark</td>
</tr>
<tr>
<td></td>
<td>Odd</td>
</tr>
<tr>
<td></td>
<td>Even</td>
</tr>
<tr>
<td>7 bits</td>
<td>Space</td>
</tr>
<tr>
<td></td>
<td>Mark</td>
</tr>
<tr>
<td></td>
<td>Odd</td>
</tr>
<tr>
<td></td>
<td>Even</td>
</tr>
<tr>
<td>8 bits</td>
<td>Space</td>
</tr>
<tr>
<td></td>
<td>Mark</td>
</tr>
<tr>
<td></td>
<td>Odd</td>
</tr>
<tr>
<td></td>
<td>Even</td>
</tr>
</tbody>
</table>

**Note:** Most serial hardware does not support the combinations 8-bit mark or 8-bit space, so these combinations of parameter values are never generated in LanguageLevel 2. (In fact, in LanguageLevel 1, these combinations behave equivalently to 8-bit None in LanguageLevel 2.)

The **CheckParity** parameter is set according to the new **Parity** setting:

- **true** if **Parity** is Odd or Even.
- **false** if **Parity** is Space or Mark.
- Not changed if **Parity** is None. (Parity checking is not done if **Parity** is independent of the setting of **CheckParity**.)

This operator should be invoked only from within a system administrator job.

**Errors:** invalidaccess, rangecheck, stackunderflow, typecheck

**See also:** sccbatch

**setsccinteractive**

```
channel baud options setsccinteractive  
```

(formerly set the values of device parameters for serial communication in interactive or emulation mode. This operator now pops its three operands off the stack and otherwise does nothing.

This operator should be invoked only from within a system administrator job.

There is no LanguageLevel 2 equivalent for this operator.)
Errors: invalidaccess, rangecheck, stackunderflow, typecheck
See also: sccinteractive

**setsoftwareiomode**  
`int setsoftwareiomode – (statusdict dictionary)`

Sets the values of the device parameters `Interpreter` and, if appropriate, `Protocol` (see Tables 3.3 and 3.4 on pages 29 and 33) in the parameter set designated by the system parameter `CurInputDevice` (Table 3.2 on page 15). The `int` operand is a code specifying the values for `Interpreter` and `Protocol`, as follows:

<table>
<thead>
<tr>
<th><code>int</code></th>
<th><code>Interpreter</code></th>
<th><code>Protocol</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PostScript</td>
<td>Normal</td>
</tr>
<tr>
<td>1</td>
<td>ProprinterXL</td>
<td>Raw</td>
</tr>
<tr>
<td>2</td>
<td>Diablo630</td>
<td>Raw</td>
</tr>
<tr>
<td>3</td>
<td>(Reserved)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>HP7475A</td>
<td>Raw</td>
</tr>
<tr>
<td>5</td>
<td>LaserJetIP or LaserJetIII (whichever is installed)</td>
<td>Raw</td>
</tr>
<tr>
<td>100</td>
<td>PostScript</td>
<td>Binary</td>
</tr>
</tbody>
</table>

*Note:* In the unlikely event that both the LaserJetIP and LaserJetIII emulators are installed, an operand value of 5 will select only LaserJetIP.

This operator should be invoked only from within a system administrator job.

Errors: invalidaccess, rangecheck, stackunderflow, typecheck
See also: softwareiomode

**settumble**  
`bool settumble – (statusdict dictionary)`

Sets the value of the page device parameter `Tumble` (PLR3, p. 416) to `bool`. If this parameter is not present, the `Policies` dictionary is consulted and the applicable recovery policy is invoked (see PLR3, Section 6.2.7).

*Note:* Because this operator invokes `setpagedevice`, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: stackunderflow, typecheck
See also: tumble
**setuserdiskpercent**  
`int setuserdiskpercent`  
(\textit{statusdict} dictionary)  

formerly set the percentage of disk space reserved for user files. This operator now pops the operand \texttt{int} off the stack and otherwise does nothing.  
This operator should be invoked only from within a system administrator job.  
There is no LanguageLevel 2 equivalent for this operator.  

Errors: invalidaccess, rangecheck, stackunderflow, typecheck  
See also: userdiskpercent

**softwareiomode**  
`– softwareiomode int`  
(\textit{statusdict} dictionary)  

returns a code representing the current value of the device parameter \texttt{Interpreter} (see Table 3.3 on page 29) in the parameter set designated by the system parameter \texttt{CurInputDevice} (Table 3.2 on page 15). See the description of the \texttt{setsoftwareiomode} compatibility operator for possible result values and their meanings.  

\textbf{Note:} If the value of \texttt{Interpreter} is not one of those listed under \texttt{setsoftwareiomode}, the \texttt{int} value returned will be -1.  

Errors: stackoverflow  
See also: setsoftwareiomode

**tumble**  
`– tumble bool`  
(\textit{statusdict} dictionary)  

returns the current value of the page device parameter \texttt{Tumble} (PLR3, p. 416). If this parameter is not present, an undefined error will occur.  

Errors: stackoverflow, undefined  
See also: settumble

**userdiskpercent**  
`– userdiskpercent int`  
(\textit{statusdict} dictionary)  

formerly returned the percentage of disk space reserved for user files. This operator now does nothing and returns the value 0.  
There is no LanguageLevel 2 equivalent for this operator.
Errors: stackoverflow
See also: setuserdiskpercent

\texttt{waittimeout} \texttt{- waittimeout \texttt{int}} \quad (\texttt{statusdict} \texttt{dictionary})

is an integer with the same value as the user parameter \texttt{WaitTimeout} (see Table 3.1 on page 14).

\texttt{waittimeout} is not an operator; it is a name in \texttt{statusdict} associated with the integer value of the wait timeout. LanguageLevel 1 applications that formerly specified the wait timeout by setting this value, using code such as

\begin{verbatim}
/waittimeout 120 def
\end{verbatim}

can continue to do so; the PostScript interpreter will automatically update the value of the \texttt{WaitTimeout} user parameter to match. Similarly, changes in the value of the \texttt{WaitTimeout} user parameter will affect the value of \texttt{waittimeout}.

Errors: stackoverflow, undefined

\texttt{11x17} \texttt{- 11x17 -} \quad (\texttt{userdict} \texttt{dictionary})

requests 11-by-17-inch paper as the output medium. It invokes the \texttt{setpagedevice} operator with a \texttt{PageSize} value of 11 by 17 inches ([792 1224] in default user space units) and an \texttt{ImagingBBox} value of \texttt{null}, denoting the largest bounding box possible for this page size on the current output device (see PLR3, p. 414).

This operator sets the \texttt{PageSize} entry in the \texttt{Policies} dictionary to 7, disabling the normal LanguageLevel 2 media selection mechanism and unconditionally establishing 11 by 17 inches as the page size whether or not an 11-by-17-inch media source is available on the device (see PLR3, pp. 434–435).

\textbf{Note: Because this operator invokes setpagedevice, it also causes all of that operator’s attendant side effects, such as reinitializing the graphics state and erasing the current page; see PLR3, pp. 679–680 and Section 6.2.6 (pp. 426–432).}

Errors: limitcheck
See also: 11x17tray
requests a paper tray containing 11-by-17-inch paper as the media source. It in-
vokes the `setpagedevice` operator with a `PageSize` value of 11 by 17 inches
([792 1224] in default user space units) and an `ImagingBBox` value of `null`, de-
noting the largest bounding box possible for this page size on the current output
device (see `PLR3`, p. 414).

This operator sets the `PageSize` entry in the `Policies` dictionary to 0, causing a
`configurationerror` to be raised if an 11-by-17-inch media source is not available
on the device (see `PLR3`, p. 434). For compatibility with LanguageLevel 1, the im-
plementation of the `11x17tray` operator converts any such `configurationerror` to a
`rangecheck` error.

Note: Because this operator invokes `setpagedevice`, it also causes all of that operator’s
attendant side effects, such as reinitializing the graphics state and erasing the current
page; see `PLR3`, pp. 679–680 and Section 6.2.6 (pp. 426–432).

Errors: limitcheck, rangecheck
See also: 11x17
APPENDIX A

Changes since Earlier Versions

A.1 Changes since Version 3010

The following changes have been made to the PostScript language since version 3010. These changes were incorporated into the 3011 release.

• A new kind of Type 0 CIDFont has been added: a CFF CIDFont, defined in FontSet resources rather than in a CIDFont file. See Section 5.1.2, “CFF CIDFonts.”

• The Type 2 CIDFont dictionary has been extended to accommodate incremental downloading of CJK TrueType CIDFonts. See Section 5.1.3, “Incremental Downloading of CJK TrueType CIDFonts.”

• A new optional entry, InterpolationDetails, has been added to all image dictionaries, regardless of type. See Section 5.2, “Interpolation Details Dictionary.”

• A new optional trapping parameter, ImagemaskTrapping, has been added to the trapping parameter dictionary. See Section 5.3, “Imagemask Trapping Parameter.”
APPENDIX B

Implementation Limits

THIS APPENDIX SUPPLEMENTS Appendix B of the PostScript Language Reference, Third Edition. It provides additional information about limits that are typical of PostScript interpreter implementations from Adobe Systems.

B.1 Columns Limit in CCITTFaxDecode

The Columns entry in the CCITTFaxDecode dictionary (PLR3, pp. 144-145) is limited to a maximum value of 62,000.
APPENDIX C

Emulator Parameter Sets

PARAMETER SETS OF TYPE Emulator provide supporting information for emulating non-PostScript imaging systems within a PostScript imaging system.

An emulator is an interpreter for input stream data that is formatted in a page description language (PDL) other than PostScript. Such a PDL would originally be used on a non-PostScript imaging system. By incorporating PDL emulators, PostScript imaging systems can emulate other imaging systems.

An imaging system needs to know how to interpret the stream of input characters on its data channels in order make marks on the page. The rules it uses are specified by the Interpreter device parameter, which is part of each Communications parameter set that is defined for the data channels. (For more information, see Section 3.3.3, “Communications Parameter Sets,” and in particular the description of the Interpreter parameter in Table 3.3 on page 29.) If the value of Interpreter is other than PostScript, the imaging system is being asked to emulate the functions of some other imaging system.

Emulator parameter sets were originally introduced to communicate to an emulation interpreter the values of job-level parameters that might change from one print job to another. They also provided a means of querying and changing the current value of a parameter. For example, a Copies parameter would indicate how many copies of a page to print; the number desired could be changed by setting the parameter.

Job-level control parameters required by non-PostScript imaging systems often varied among products using the same interpreter PDL. As a result, the term emulator came to refer, in the context of emulator parameter sets, to the imaging system itself, rather than to the interpreter PDL.
This situation changed when PCL (Printer Control Language) interpreter imaging systems introduced the Printer Job Language (PJL) to support job-level control parameters. PJL replaced the need to rely on emulator parameter sets for this purpose. However, emulator parameter sets are still useful in some cases. A particular imaging system product might use them for its own emulator purposes. They might also be used to share a parameter between PostScript and an emulator.

As a side effect of the introduction of PJL, the use of the term emulator also changed (in the context of Emulator parameter sets) from referring to a particular imaging system to referring to the interpreter PDL being emulated. Note that with the introduction of the PDL resource category, any desired information about a specific imaging system target or version of a PDL being emulated is appropriately provided using this resource.

PCL is an emulator that is included in many PostScript imaging systems. The %PCL% parameter set applies to systems that include an emulator for one or more versions of PCL. The required entries in the %PCL% parameter set are described in Table C.1.

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td><em>(Read-only)</em> The parameter set type; must be Emulator.</td>
</tr>
<tr>
<td>MaxPermanentStorage</td>
<td>integer</td>
<td>The maximum amount of memory (in bytes) to be dedicated for use by the PCL emulator.</td>
</tr>
</tbody>
</table>
APPENDIX D

Typical Font Set

THIS APPENDIX SUPPLEMENTS Appendix E of the PostScript Language Reference, Third Edition. It lists the typical set of fonts available on most Adobe PostScript 3 products. While there is no standard set of required fonts, Adobe PostScript 3 products typically have the 136 fonts listed below built into the interpreter.

AlbertusMT
AlbertusMT-Italic
AlbertusMT-Light
AntiqueOlive-Roman
AntiqueOlive-Italic
AntiqueOlive-Bold
AntiqueOlive-Compact
Apple-Chancery
ArialMT
Arial-ItalicMT
Arial-BoldMT
Arial-BoldItalicMT
AvantGarde-Book
AvantGarde-BookOblique
AvantGarde-Demi
AvantGarde-DemiOblique
Bodoni
Bodoni-Italic
Bodoni-Bold
Bodoni-BoldItalic
Bodoni-Poster
Bodoni-PosterCompressed
Bookman-Light

Bookman-LightItalic
Bookman-Demi
Bookman-Demilight
Carta
Chicago
Clarendon
Clarendon-Light
Clarendon-Bold
CooperBlack
CooperBlack-Italic
Copperplate-ThirtyTwoBC
Copperplate-ThirtyThreeBC
Coronet-Regular
Courier
Courier-Oblique
Courier-Bold
Courier-BoldOblique
Eurostile
Eurostyle-Bold
Eurostyle-ExtendedTwo
Eurostyle-BoldExtendedTwo
Geneva
GillSans
<table>
<thead>
<tr>
<th>Font Style</th>
<th>Font Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GillSans-Italic</td>
<td>LubalinGraph-Demi</td>
</tr>
<tr>
<td>GillSans-Bold</td>
<td>LubalinGraph-DemiOblique</td>
</tr>
<tr>
<td>GillSans-BoldItalic</td>
<td>Marigold</td>
</tr>
<tr>
<td>GillSans-Condensed</td>
<td>Monaco</td>
</tr>
<tr>
<td>GillSans-BoldCondensed</td>
<td>MonaLisa-Recut</td>
</tr>
<tr>
<td>GillSans-Light</td>
<td>NewCenturySchlbk-Roman</td>
</tr>
<tr>
<td>GillSans-LightItalic</td>
<td>NewCenturySchlbk-Italic</td>
</tr>
<tr>
<td>GillSans-ExtraBold</td>
<td>NewCenturySchlbk-Bold</td>
</tr>
<tr>
<td>Goudy</td>
<td>NewCenturySchlbk-BoldOblique</td>
</tr>
<tr>
<td>Goudy-Italic</td>
<td>NewYork</td>
</tr>
<tr>
<td>Goudy-Bold</td>
<td>Optima</td>
</tr>
<tr>
<td>Goudy-BoldItalic</td>
<td>Optima-Italic</td>
</tr>
<tr>
<td>Goudy-ExtraBold</td>
<td>Optima-Bold</td>
</tr>
<tr>
<td>Helvetica</td>
<td>Optima-BoldItalic</td>
</tr>
<tr>
<td>Helvetica-Oblique</td>
<td>Oxford</td>
</tr>
<tr>
<td>Helvetica-Bold</td>
<td>Palatino-Roman</td>
</tr>
<tr>
<td>Helvetica-BoldOblique</td>
<td>Palatino-Italic</td>
</tr>
<tr>
<td>Helvetica-Condensed</td>
<td>Palatino-Bold</td>
</tr>
<tr>
<td>Helvetica-Condensed-Oblique</td>
<td>Palatino-BoldItalic</td>
</tr>
<tr>
<td>Helvetica-Condensed-Bold</td>
<td>StempelGaramond-Roman</td>
</tr>
<tr>
<td>Helvetica-Condensed-BoldObl</td>
<td>StempelGaramond-Italic</td>
</tr>
<tr>
<td>Helvetica-Narrow</td>
<td>StempelGaramond-Bold</td>
</tr>
<tr>
<td>Helvetica-Narrow-Oblique</td>
<td>StempelGaramond-BoldOblique</td>
</tr>
<tr>
<td>Helvetica-Narrow-Bold</td>
<td>Symbol</td>
</tr>
<tr>
<td>Helvetica-Narrow-BoldOblique</td>
<td>Tekton</td>
</tr>
<tr>
<td>HoeflerText-Regular</td>
<td>Times-Roman</td>
</tr>
<tr>
<td>HoeflerText-Italic</td>
<td>Times-Italic</td>
</tr>
<tr>
<td>HoeflerText-Black</td>
<td>Times-Bold</td>
</tr>
<tr>
<td>HoeflerText-BlackItalic</td>
<td>Times-BoldItalic</td>
</tr>
<tr>
<td>HoeflerText-Ornaments</td>
<td>TimesNewRomanPSMT</td>
</tr>
<tr>
<td>JoannaMT</td>
<td>TimesNewRomanPS-ItalicMT</td>
</tr>
<tr>
<td>JoannaMT-Italic</td>
<td>TimesNewRomanPS-BoldMT</td>
</tr>
<tr>
<td>JoannaMT-Bold</td>
<td>TimesNewRomanPS-BoldItalicMT</td>
</tr>
<tr>
<td>JoannaMT-BoldItalic</td>
<td>Univers</td>
</tr>
<tr>
<td>JoannaMT-BoldOblique</td>
<td>Univers-Oblique</td>
</tr>
<tr>
<td>LetterGothic</td>
<td>Univers-Oblique</td>
</tr>
<tr>
<td>LetterGothic-Slanted</td>
<td>Univers-Bold</td>
</tr>
<tr>
<td>LetterGothic-Bold</td>
<td>Univers-BoldOblique</td>
</tr>
<tr>
<td>LetterGothic-BoldSlanted</td>
<td>Univers-Light</td>
</tr>
<tr>
<td>LubalinGraph-Book</td>
<td>Univers-LightOblique</td>
</tr>
<tr>
<td>LubalinGraph-BookOblique</td>
<td>Univers-Condensed</td>
</tr>
<tr>
<td>Font Style</td>
<td>Font Style</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Univers-CondensedOblique</td>
<td>Univers-BoldExt</td>
</tr>
<tr>
<td>Univers-CondensedBold</td>
<td>Univers-BoldExtObl</td>
</tr>
<tr>
<td>Univers-CondensedBoldOblique</td>
<td>Wingdings-Regular</td>
</tr>
<tr>
<td>Univers-Extended</td>
<td>ZapfChancery-MediumItalic</td>
</tr>
<tr>
<td>Univers-ExtendedObl</td>
<td>ZapfDingbats</td>
</tr>
</tbody>
</table>
APPENDIX E

Updates to the *PostScript Language Reference*, Third Edition

This appendix describes changes, additions, and errata to the *PostScript Language Reference*, Third Edition. The changes described here apply to all implementations of LanguageLevel 3, and are limited to additions, corrections, and clarifications to material in PLR3; they do not include any changes in the PostScript language that have been introduced as extensions to LanguageLevel 3.

E.1 Errata for the First Printing

This section gives errata (not yet published) to the first printing of the *PostScript Language Reference*, Third Edition. This is the only printing that has appeared up to the time of publication of this Supplement.

E.1.1 Chapter 4 Errata

The following change should be made in Chapter 4:

- Page 212, Figure 4.5. There is an extraneous horizontal line between CIEBased-DEFG and DeviceRGB.

E.1.2 Chapter 5 Errata

The following change should be made in Chapter 5:

E.1.3 Chapter 6 Errata

The following changes should be made in Chapter 6:

• Page 408, last line before the code example. Change “a sheet of special media” to “a sheet of special medium.”

• Page 427. Insert following the first paragraph after the table:

  Note: The page device’s default Install procedure establishes the default device-dependent graphics state parameters that are appropriate for the device. If a PostScript program replaces the Install procedure, the replacement procedure should call the original procedure as part of its execution; otherwise, some parameters may not be initialized properly.

• Pages 447–449, Table 6.13. The following entry was omitted from the table:

<table>
<thead>
<tr>
<th>KEY</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HalftoneName</td>
<td>name or null</td>
<td>(Optional) The name of a Halftone resource with which to mark traps. It is recommended that the resource be defined in global VM. The Halftone resource may be used at unpredictable times, including at a save level lower than the current one. If this parameter is null or undefined, traps will be marked using the halftone in effect at the time, possibly causing unexpected results.</td>
</tr>
</tbody>
</table>

E.1.4 Chapter 8 Errata

The following changes should be made in Chapter 8:

• Page 552, cshow. Add stringwidth and charpath to the “See Also” list. (This is to indicate that the special glyph selection process applies to these operators as well as to the ones having the word show in their names.)

• Page 557, currentfont. Append the following paragraph:

  Note: As discussed in Section 5.10.3, “Nested Composite Fonts,” the makefont, scale-font, and selectfont operators do not apply their transformations recursively to the descendant base fonts or CIDFonts of a composite font. When the current font is such an untransformed descendant of a transformed composite font, currentfont returns a copy of the descendant font whose FontMatrix has been appropriately transformed to reflect the transformation previously performed on the parent composite font.
• Page 620, languagelevel. Change “returns an integer designating…” to “is an integer designating…” At the end of the operator description, add the sentence “languagelevel is not an operator; it is a name in systemdict associated with the integer.”

• Page 716, version. Change “returns a string that identifies…” to “is a read-only string object that identifies…” At the end of the operator description, add the sentence “version is not an operator; it is a name in systemdict associated with the string.”

E.1.5 Bibliography Errata

The following changes should be made in the Bibliography:

• Page 811, following first paragraph. The correct URL for the Adobe Developer Relations Web site is now:

  <http://partners.adobe.com/asn/developer/>

• Page 814, fifth paragraph. Change to read as follows:

  International Organization for Standardization (ISO), ISO/IEC 10918-1, Digital Compression and Coding of Continuous-Tone Still Images. Informally known as the JPEG standard, for the Joint Photographic Experts Group (the organization that developed the standard). May be ordered from ISO at <http://www.iso.ch>.
Bibliography

SOME DOCUMENTS LISTED IN THIS BIBLIOGRAPHY are indicated as being available on the Adobe Developer Relations site on the World Wide Web. This site is located at

<http://partners.adobe.com/asn/developer/>

Document version numbers given in this Bibliography are the latest as of the time of publication; more recent versions may be found on the Web site.

Resources from Adobe Systems Incorporated


*PostScript Language Program Design*, Addison-Wesley, Reading, MA, 1988. Though this edition of this book describes LanguageLevel 1 only, it is still useful for programmers interested in the effective and efficient design of PostScript programs and printer drivers.

Technical Notes. The following Technical Notes are available on the Adobe Developer Relations Web site:

- *Adobe Serial and Parallel Communications Protocols Specification*, Technical Note #5009

Other Resources


Index

A
access attributes  61
accuratescreens compatibility operator  94
AccurateScreens user parameter  14, 94, 113
ActualAngle entry (type 1 halftone dictionary)  97
ActualFrequency entry (type 1 halftone dictionary)  97
Address device parameter  44, 54
Adobe Developer Relations (URL)  137
Apple® Computer, Inc.  22
AppleTalk  21, 37, 38, 42, 43, 44, 47, 94
appletalktype compatibility value  43, 44, 94
%AppSocket% parameter set  38
(block) 40–41
AppSocket protocol  38
a3 compatibility operator  92
a3tray compatibility operator  92
a4 compatibility operator  93
a4small compatibility operator  93, 108
a4tray compatibility operator  94

B
bands (raster memory)  15, 17
Baud device parameter  35, 112, 119
Bind page device parameter  9, 10
BindDetails page device parameter  10, 11
BlockSize device parameter  62, 63, 65, 66, 67
Boise specification (Hewlett-Packard)  36
Booklet page device parameter  9
BookletDetails page device parameter  9, 11
BootDelay device parameter  55, 56
BOOTP sequence  50
Bridging device parameter  44, 54
BroadcastAddress device parameter  50, 51
BSizeStandard device parameter  56
buildtime compatibility value  96
BuildTime system parameter  15, 96
Bus device parameter  37, 61
byteorder compatibility value  96
ByteOrder system parameter  15, 96
b5 compatibility operator  95
b5tray compatibility operator  95

C
%Calendar% parameter set  46
(table) 60
calibrationDateTimeTag entry (ICC profile)  82, 85
%cartridge% parameter set  60
(table) 64–65
CartridgeID device parameter  65
CartridgeType device parameter  64
CCITTFaxDecode filter parameter dictionary  127
CDevProc entry (Type 2 CIDFont dictionary)  80, 81
CFF (Compact Font Format)  75, 76, 77
  CIDFonts  77, 125
Chameleons fonts  75, 76, 77
character sets  57–58, 72
charpath operator  136
CharSet device parameter  57
CharSet entry (Localization resource)  72
CheckParity device parameter  35, 36, 55, 119, 120
checkpassword compatibility operator  96
checkscreen compatibility operator  97
Checksum device parameter  49, 52
CIDCount entry (Type 2 CIDFont dictionary)  78
CIDFonts
  CFF  77, 125
  TrueType, CJK  78–81, 125
  Type 0  77, 125
  Type 2  78, 79, 125
CIDFontType entry (CIDFont dictionary)  77
CID-keyed fonts  78
CIDMap entry (Type 2 CIDFont dictionary)  78–79
CJK (Chinese, Japanese, and Korean) TrueType
  CIDFonts  78–81, 125
Collate page device parameter  8, 9, 10
CollateDetails page device parameter  8, 10–11
  typical entries (table) 11
color rendering dictionaries 82–85
  ColorantType entry (colorant subdictionary) 82
  ColorBurst hardware rendering device 17
  ColorRendering resource category 84, 85
  Columns entry (CCITTFaxDecode filter parameter dictionary) 127
  %CommName% parameter sets 23–28
  %CommName_NV% parameter sets 23–28
  %CommName_Pending% parameter sets 23–28
  Communications parameter sets 15, 19, 21–44, 45, 129
    entries common to all (table) 29–32
    point-to-point 33–37
    setting up 23–28
  See also individual parameter sets by name
  communications protocols 33–34
  compatibility operators 87–124
    (summary) 88–91
  See also individual operators by name
  composite fonts 136
  CompressImageSource system parameter 15
  compression filters 15
  configuration error 3
  configurationerror error 20, 29, 31, 33, 34, 36, 37, 40, 41
    (summary) 88–91
  See also individual operators by name
  ConnectorType device parameter 54
  %Console% parameter set 46, 71
    (table) 57–58
  ControlLanguage resource category 69, 70
    (table) 70
  ControlPortNumber device parameter 40, 41
  copypage operator 9, 10, 106
  country codes 58, 71
    (table) 59
  Country device parameter 58
  Country entry (Localization resource) 71, 72
  crdInfoTag entry (ICC profile) 83–85
    format (table) 83–84
  CRDs. See color rendering dictionaries
  CreationDate entry (color rendering dictionary) 82–85
  cshow operator 136
  CurBufferType system parameter 15
  CurDisplayList system parameter 15
  CurFontCache system parameter 15
  CurFormCache system parameter 15
  CurlInputDevice system parameter 15, 102, 121, 122
  CurOutlineCache system parameter 15
  CurOutputDevice system parameter 16
  CurPatternCache system parameter 15
  current input file 15
  current output file 16
  currentdevparams operator 19, 61
  currentfont operator 136
  CurScreenStorage system parameter 15
  CurSourceList system parameter 15
  CurStoredScreenCache system parameter 15
  CurUPathCache system parameter 15

D
  Darkness device parameter 56, 57
  DarknessBlack device parameter 57
  DarknessCyan device parameter 57
  DarknessMagenta device parameter 57
  DarknessYellow device parameter 57
  DARPA (Defense Advanced Research Projects Agency) 22
  DataBits device parameter 35, 112–113, 119–120
  DataPortNumber device parameter 40, 41
  DataSource dictionary entry 18
  Day device parameter 60
  DDP (Datagram Delivery Protocol) 22, 47
  defaulttimeouts compatibility operator 97
  DelayedOutputClose device parameter 30
  details dictionaries 3–4
  devdismount compatibility operator 97
  devforall compatibility operator 98
  devformat compatibility operator 98
  device parameters 18–67
    Communications 15, 19, 21–44, 45, 112–113, 119–120, 129
    Emulator 19, 129–130
    FileSystem 19, 60–67
    interdependencies 20
    Parameters 19, 21, 38, 40, 44–60
    See also individual parameters and parameter sets by name
  DeviceCMY process color model 110
  DeviceCMYK process color model 110
  DeviceGray process color model 110
  DeviceN process color model 110
  DeviceRenderingInfo page device parameter 6–7
    typical entries (table) 7
  DeviceRGB process color model 110
  DeviceRGBK process color model 110
  devmount compatibility operator 98
  devstatus compatibility operator 99
  directory service advertisements (SAs) 42
%disk% parameter set 60, 99, 102
  (table) 61–64
diskonline compatibility operator 99
diskstatus compatibility operator 99
display lists, memory allocation for 17
dopportinters compatibility operator 100
doPrintErrors system parameter 16, 100, 114
dostartpage compatibility operator 100
doStartPage system parameter 16, 100, 114
dosysstart compatibility operator 100
DSC (Document Structuring Conventions) 77
DTR flow control 34, 112, 119
duplexmode compatibility operator 100
duplexmode page device parameter 100, 107–108, 115
duplexmode device parameter 100
DUPLEX page device parameter 8, 9
duplexorientation page device parameter 11
DST (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUPLEX page device parameter 8, 9
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
duplexmode device parameter 37, 39, 40, 41, 42, 43
Duplex page device parameter 100, 107–108, 115
DUS (Document Structuring Conventions) 77
%EtherTalk% parameter set 39, 94, 115
  (table) 43
EtherTalkType device parameter 43, 44, 94, 95
EtherTalkZone device parameter 43
ETX/ACK flow control 34, 112, 119
ExitJamRecovery page device parameter 11
exitserver parameter 16, 96

F
FactoryDefaults system parameter 15
FatalErrorAddress system parameter 16
FDAArray subdictionaries (Type 0 CIDFont dictionary) 77
file server 41
FileSystem parameter sets 19, 60–67
  See also individual parameter sets by name
filter operator 73
Filter resource category 74
Filtering device parameter 37, 39, 40, 41, 42, 43
  filters, compression 15
firstside compatibility operator 101
FlowControl device parameter 34, 112, 119
Fold page device parameter 8, 9
FoldDetails page device parameter 9, 11
font dictionaries 77
Font resource category 77
FontMatrix entry (CIDFont dictionary) 136
FontResourceDir system parameter 15, 76
fonts 75–81
  CFF (Compact Font Format) 75, 76, 77
  Chameleon 75, 76, 77
  CIDFonts 77, 78–81, 125
  CID-keyed 78
  composite 136
  TrueType 78
  Type 0 CIDFonts 77, 125
  Type 1 77
  Type 2 75, 77
  Type 2 CIDFonts 78, 79, 125
  Type 14 75, 77
  Type 42 78
  typical set 131–133
%fontset% fictitious file system 75–77
FontSet resource category 76, 77, 125
FontType entry
  FDAArray subdictionary 77
  font dictionary 77
frame buffer 15
Free device parameter 62, 65, 66, 99
Index

**G**
- **GatewayAddress** device parameter 50, 51
- **GenericResourceDir** system parameter 15
- **GenericResourcePathSep** system parameter 15
- **GIFDecode** filter 74
- glyph indices 78–79
- glyph metrics 79–81
- **GlyphDirectory** entry (Type 2 CIDFont dictionary) 78, 79–81
- **Groups** device parameter 42

**H**
- **Halftone** resource category 136
- **HalftoneMode** user parameter 14
- **HalftoneName** trapping parameter 136
- **Handshake** device parameter 36, 37
- hardware rendering devices 17
- **hardwareiomode** compatibility operator 102
- **HashNames** device parameter 29, 61, 62, 65, 66, 98, 99
- height (glyph) 80, 81
- Hewlett-Packard Company 31
- Boise specification 36
- “hmtx” table (TrueType font) 79, 80, 81
- **HopCount** device parameter 53
- host (definition) 13
- **Hour** device parameter 60
- **HTML** (Hypertext Markup Language) 74
- **HWOptions** resource category 69, 73
- **HWResolution** page device parameter 111, 119

**I**
- IBM. See International Business Machines Corporation
- IBM Personal Computer (PC) 31, 34, 36
- **ICC** (International Color Consortium) profiles 82–85
- **IDE** (Integrated Drive Electronics) interface 44, 56, 61
- %ide% parameter set 46, 61
- **Imaging** system (definition) 13
- **ImagingBBox** page device parameter 92, 93, 94, 95, 103, 104, 105, 106, 123, 124
- implementation limits 127
- implicit resources 73–74
- See also individual resource categories by name
- incremental downloading (CJK TrueType CIDFonts) 78–81, 125
- initialization, system 16, 17, 18, 55, 56
- **InitializeAction** device parameter 63, 64, 65, 66, 67, 98, 102
- **InitiatorId** device parameter 55
- input file, current 15
- **InsertSheet** page device parameter 10
- **Install** procedure 11, 136
- **InstalledRam** system parameter 17
- **Interleave** device parameter 63
- International Business Machines Corporation (IBM) 31
- International Color Consortium (ICC) 82
- **InterpolationDetails** entry (image dictionary) 81, 125
- **Interpreter** device parameter 20, 31, 32, 33, 34, 71, 121, 122, 129
- interpreter parameters 13–67
- **invalidaccess** error 101
- **IODevice** resource category 19, 73, 74, 98
- **ioerror** error 36, 63
- **IP** (Internet Protocol) 21, 22, 38, 40, 45, 46, 47, 49
- **IPX** (Internetwork Packet Exchange) protocol 21, 22, 38, 39, 43, 44, 46, 47, 52
- %IP% parameter set 46
- (table) 49–52
- **IPAddress** device parameter 50, 51, 52
- **IPAddressDynamic** device parameter 50, 51, 52
- **IPX** (Internetwork Packet Exchange) protocol 21, 22, 38, 39, 43, 44, 46, 47, 52
- %IPX% parameter set 46
- (table) 52–53
- **ISO** (International Organization for Standardization) 56, 58, 95, 96, 137

**J**
- **Japanese Font Downloader utility** 64
- **JIS** (Japanese Industrial Standards Committee) 56, 95, 96
- **jobname** compatibility value 102
- **JobName** user parameter 14, 102
Index

JobTimeout  
system parameter 17, 97, 113  
user parameter 14, 17, 103, 115  
jobtimeout compatibility operator 103  
JPEG (Joint Photographic Experts Group) standard 137

K

KeepaliveTimer device parameter 42

L

Laminate page device parameter 10  
language codes 58, 71  
(table) 58  
Language device parameter 58  
Language entry (Localization resource) 71, 72  
LanguageFamily entry  
  ControlLanguage resource 70  
  PDL resource 71  
LanguageLevel entry  
  ControlLanguage resource 70  
  PDL resource 71  
languagelevel value (systemdict dictionary) 137  
LanguageVersion entry  
  ControlLanguage resource 70  
  PDL resource 71  
LaserJet printer 31, 32  
LaserJet 4 emulation 16, 31, 32  
LaserWriter printer 43  
%LAT% parameter set 38  
  (table) 42  
LAT (Local Area Transport) protocol 38  
ledger compatibility operator 103  
ledgertray compatibility operator 103  
left sidebearing (glyph) 80, 81  
legal compatibility operator 104  
legaltray compatibility operator 104  
letter compatibility operator 105  
lettersmall compatibility operator 105, 108  
lettertray compatibility operator 106  
LicenseID system parameter 15  
Limit entry (DeviceRenderingInfo dictionary) 7  
LLAP (LocalTalk Link Access Protocol) 22  
Localization resource category 69, 71–73  
  naming guidelines 72–73  
  (table) 71–72  
LocalTalk 20, 39  
%LocalTalk% parameter set 23, 39, 94, 102, 115  
  (table) 42–43  
LocalTalkType device parameter 42, 43, 44, 94, 95  
LogHost device parameter 48  
LogicalSize device parameter 62, 63, 65, 66, 67, 98, 99, 102  
LoginPassword device parameter 41  
LogPriority device parameter 48  
lpr command (UNIX) 38, 39  
%LPR% parameter set 38  
  (table) 39–40  
Macintosh® computer 22, 37  
makefont operator 136  
manualfeed compatibility value 106  
ManualFeed page device parameter 106  
manualfeedtimeout compatibility value 107  
ManualFeedTimeout page device parameter 5, 97, 107, 113  
margins compatibility operator 107  
Margins page device parameter 107, 116  
MaxDictStack user parameter 14  
MaxDisplayAndSourceList system parameter 15  
MaxDisplayList system parameter 15  
MaxExecStack user parameter 14  
MaxFontCache system parameter 15  
MaxFontItem user parameter 14  
MaxFormCache system parameter 15  
MaxFormItem user parameter 14  
MaxHWRenderingBuffer system parameter 17  
MaxImageBuffer system parameter 15  
MaxLocalVM user parameter 14  
MaxOpStack user parameter 14  
MaxOutlineCache system parameter 15  
MaxPatternCache system parameter 15  
MaxPatternItem user parameter 14  
MaxPermanentStorage device parameter 130  
MaxPermanentVM system parameter 16, 17  
MaxRasterMemory system parameter 15, 17  
MaxScreenItem user parameter 14  
MaxScreenStorage system parameter 15  
MaxSourceList system parameter 15  
MaxStoredScreenCache system parameter 15  
MaxSuperscreen user parameter 14  
MaxUPathCache system parameter 15  
MaxUPathItem user parameter 14
media selection, page device parameter for 5
   (table) 5
Metrics entry (Type 2 CIDFont dictionary) 80, 81
MetricsCount entry (Type 2 CIDFont dictionary) 79–81 values (table) 81
Metrics2 entry (Type 2 CIDFont dictionary) 79–81
MIB (Management Information Base) 45
MinBandBuffers system parameter 17
MinFontCompress user parameter 14
Minute device parameter 60
MirrorPrint compatibility operator 107
MirrorPrint page device parameter 107, 116
Month device parameter 60
Mounted device parameter 61, 62, 65, 66, 97, 98, 99
MulticastTimer device parameter 42

N
nAckPulseWidth device parameter 37
Name device parameter 54
naming conventions (parameter sets) 23–24, 45
natural languages 46, 58, 69, 71–73
network communications 21
NetworkAddress device parameter 53
NetworkMask device parameter 50, 51
newsheet compatibility operator 107
node address 47, 48, 53
   forms (table) 47
node number, Novell 47, 53, 54
NodeID device parameter 42, 43, 44
   nonvolatile memory 18, 24, 26
note compatibility operator 108
Novell NetWare 21, 22, 38, 41, 46, 53, 54
   node number 47, 53, 54
Novell, Inc. 38
nStrobeExpectedPulseWidth device parameter 37

O
On device parameter 20, 29, 35, 49, 52, 54, 115
   operators, compatibility 87–124
   (summary) 88–91
   See also individual operators by name
Orientation page device parameter 109, 116–117, 117–118
os device 61
   %os% parameter set 61
   (table) 66–67
OSI (Open Systems Interconnection) model 19, 21, 38, 44
output file, current 16
OutputDevice device parameter 36
OutputFaceUp page device parameter 110, 118
OutputPage page device parameter 8

P
page delivery, page device parameters for 8–11
   (table) 8–10
page description languages (PDL) 16, 31, 32, 37, 70–71,
   129, 130
page device parameters 3–12
   categories (table) 4–5
   See also individual parameters by name
page image placement, page device parameters for 5–6
   (table) 5–6
page sets 9, 10
PageCount
   device parameter 57
   system parameter 15, 108
pagecount compatibility operator 108
pagemargin compatibility operator 109
PageOffset page device parameter 109, 117–118
pageparams compatibility operator 109
PageSize page device parameter 12, 92, 93, 94, 95, 103,
   124
pagestackorder compatibility operator 110
parallel channel 21
%Parallel% parameter set 23, 102, 115
   (table) 36–37
parameter sets, device 19
%CommName% 23–28
%CommName_NV% 23–28
%CommName_Pending% 23–28
Communications 15, 19, 21–44, 45, 129
Emulator 19, 129–130
FileSystem 19, 60–67
naming conventions 23–24, 45
Parameters 19, 21, 38, 40, 44–60
point-to-point 33–37
   See also individual parameter sets by name
parameters
device 18–67, 112–113, 119–120
   interpreter 13–67
   page device 3–12
   system 15–18
   user 14
   See also individual parameters by name
See also individual parameter sets by name

**Parameters** parameter sets 19, 21, 38, 40, 44–60

**Parity** device parameter 35, 36, 112–113, 119–120

**Password** entry (Localization resource) 73

**PC.** See IBM Personal Computer

**PCL (Printer Control Language)** 31, 32, 130

**PDL.** See page description languages

**PDL resource category** 31, 69, 70–71, 74, 130

**permanent virtual memory (VM)** 17

**PERSONALITY variable** (PJL) 32

**PhoneNET** 22, 39

**Physical** device parameter 42, 49, 50, 52, 53

**PhysicalSize** device parameter 62, 63, 65, 66, 67

**PixelBurst hardware rendering device** 17

**PJL (Printer Job Language)** 16, 32, 33, 34, 70

**PNGDecode** filter 74

**point-to-point Communications parameter sets** 33–37

**point-to-point connections** 21


**Poll** device parameter 55, 56

**port number** 40–41

**PostRenderingEnhance** page device parameter 6, 7

**PostRenderingEnhanceDetails** page device parameter 6, 7

**typical entries** (table) 8

**PostScript® printer description (PPD)** 84, 85

**PD.** See PostScript® printer description

**PreferredServer** device parameter 41, 53

**PrepareAction** device parameter 64

**PreRenderingEnhance** page device parameter 6, 7

**PreRenderingEnhanceDetails** page device parameter 6, 7

**PrinterControl** device parameter 31, 32, 33, 34, 70

**printename compatibility operator** 110

**PrinterName** system parameter 15, 42, 43, 44, 110, 118

**PrintHost** device parameter 39, 40

**PrintQuality** entry (**PostRenderingEnhanceDetails** dictionary) 8

**printscreen jobs** 31

**%PrintServer% parameter set** 38, 53

**(table) 41

**Private** entry (Type 1 font dictionary) 77

**PrivateHost** device parameter 47

**process color model** 110

**processcolors** compatibility operator 110

**product**

compatibility value 110

**operator** 85, 110

**profile registry** 84, 85

**profiles, ICC** 82–85

**ProofSet** entry (**CollateDetails** dictionary) 11

**Protocol** device parameter 20, 31, 33, 34, 35, 36, 37, 121

**protocols, communications** 33–34

**Punch** page device parameter 10

**PunchDetails** page device parameter 10, 11

**R**

**%ram% parameter set** 60, 61

**(table) 65–66

**RAM.** See random-access memory

**ramsize** compatibility operator 111

**RamSize** system parameter 17, 111

**random-access memory (RAM)** 17–18, 24, 25, 60, 111

**rangecheck error** 39, 48, 50, 51, 92, 94, 96, 101, 103, 104, 106, 124

**RARP sequence** 50

**raster memory** 15, 17

**read-only memory (ROM)** 60

**realformat** compatibility value 111

**RealFormat** system parameter 15, 111

**ReceiveWindowSize** device parameter 40, 49, 52

**regular resources** 69–73

See also individual resource categories by name

**%RemotePrinter% parameter set** 38

**(table) 41

**Removable** device parameter 62, 65, 66, 99

**rendering control, page device parameters for** 6–8

**(table) 6

**rendering devices, hardware** 17

**rendering intents** 83, 84, 85

**resolution** compatibility operator 111

**resourceforall operator** 98

**resources**

implicit 73–74

regular 69–73

See also individual resource categories by name

**resourcestatus operator** 77

**restore operator** 14

**RetransmitLimit** device parameter 42

**RetransmitTimer** device parameter 42

**ReusableStreamDecode** filter 60
Index

REValue entry (PostRenderingEnhanceDetails dictionary) 8
revision compatibility value 111
Revision system parameter 15, 111
RIF (Routing Information Field) 44
RIP (Routing Information Protocol) 51
%rom% parameter set 60
(table) 64–65
ROM. See read-only memory
Running device parameter 60

S
save operator 14, 17, 136
scalefont operator 136
sccbatch compatibility operator 112
sccinteractive compatibility operator 113
SCSI (Small Computer Systems Interface) 21, 44, 46, 55, 61
bus address 55
%Scsi% parameter set 46, 61
(table) 55
%ScsiComm% parameter set 55
(table) 37
Searchable device parameter 62, 65, 66, 76, 99
SearchOrder device parameter 62, 65, 66, 76, 99
Second device parameter 60
selectfont operator 136
Selector entry
ControlLanguage resource 70
PDL resource 71
SendWindowSize device parameter 39, 40, 49
serial channel 20, 21
%Serial% parameter set 15, 19, 23, 102, 115
(table) 33–36
%Serial_NV% parameter set 112, 119
%SerialB% parameter set 23, 102, 115
%SerialB_NV% parameter set 112, 119
service advertisements (SAs) 42
setaccuratescreens compatibility operator 113
setdefaulttimeout compatibility operator 113
setdevparams operator 19, 31, 34, 47, 48, 50, 51, 52, 61, 64, 76, 87
setdoprinters compatibility operator 114
setdostartpage compatibility operator 114
setdosysstart compatibility operator 114
setduplexmode compatibility operator 115
sethardwareiomode compatibility operator 115

setjobtimeout compatibility operator 115
setmargins compatibility operator 116
setmirrorprint compatibility operator 116
setpage compatibility operator 116
setpagemargin compatibility operator 117
setpageparams compatibility operator 117
setpagestackorder compatibility operator 118
setprintername compatibility operator 118
setresolution compatibility operator 119
setscchatch compatibility operator 119
setsccinteractive compatibility operator 120
setscreen operator 97
setsoftwareiomode compatibility operator 121, 122
setsystemparams operator 87
settumble compatibility operator 121
setuserdiskpercent compatibility operator 122
sfnts entry (Type 2 CIDFont dictionary) 79
showpage operator 9, 10, 106
Signature page device parameter 5, 6
SignatureDetails page device parameter 6
SlipSheet page device parameter 10
SNAP header 50, 52
SNMP (Simple Network Management Protocol) 44, 45, 47
%SNMP% parameter set 45
(table) 47–48
softwareiomode compatibility operator 122
SourceListBypass system parameter 18
Speed device parameter 54
spooler 13, 23, 25, 26, 28, 43
SPX (Sequenced Packet Exchange) protocol 21, 22, 46, 47, 52
%SPX% parameter set 46
(table) 52
Staple page device parameter 3, 9
StapleDetails page device parameter 3, 9, 11
start page 16
startjob operator 96
StartJobPassword system parameter 15, 96–97
startup action 100
StartupMode system parameter 15, 100, 114
status operator 77
statusdict dictionary 87, 94, 96, 102, 106, 107, 110, 111, 123
StatusPortNumber device parameter 40, 41
%stderr file 36, 38
%stdout file 36, 38
stencil masks 82
StopBits device parameter 35, 112, 119
stringWidth operator 136
SunOS™ 48
SysContact device parameter 47
SysLocation device parameter 47
Syslog facility 38, 44, 45, 48
%Syslog% parameter set 45

StopBits device parameter 47
system initialization 16, 17, 18, 55, 56
system parameters 15–18

SystemParamsPassword system parameter 15, 73, 96–97

T

TargetId device parameter 55
TBCP (Tagged Binary Communication Protocol) 31, 33, 37
TCP (Transmission Control Protocol) 21, 22, 38, 40, 45, 46, 47, 49
%TCP% parameter set 40, 45
(time) 49
%Telnet% parameter set 38
timeout error 5, 14
timeout intervals
job 14, 17
wait 14, 18, 123

TimeToStandby device parameter 57
TLAP (TokenTalk Link Access Protocol) 22
TokenRing 21, 22, 39, 44, 46, 53, 54
%TokenRingPhysical% parameter set 46, 50, 52, 53
(time) 54
TokenTalk 22, 38, 39, 43, 44
%TokenTalk% parameter set 39
(time) 43–44

TokenTalkType device parameter 43, 44

TonerSaver entry (PostRenderingEnhanceDetails dictionary) 8
top sidebearing (glyph) 80, 81
TranScript® 38
TransmitEncapsulation device parameter 49, 52

TrapHost device parameter 48
Trim page device parameter 9
TrueType fonts 78
CJK CIDFonts 78–81, 125
tumble compatibility operator 122
Tumble page device parameter 121, 122

Type entry
%Calendar% parameter set 60
%Cartridge% parameter set 64
CollateDetails dictionary 10, 11
Communications parameter sets 29
%ConsoLe% parameter set 57
details dictionaries 3–4
device parameter sets 19
DeviceRenderingInfo dictionary 6–7
%Disk% parameter set 61
%Engine% parameter set 56
%EthernetPhysical% parameter set 54
%Engine% parameter set 56

InterpolationDetails dictionary 81
%IP% parameter set 49
%IPX% parameter set 52
%os% parameter set 66
%PCL% parameter set 130

PostRenderingEnhanceDetails dictionary 7, 8
PreRenderingEnhanceDetails dictionary 7

%ram% parameter set 65
%rom% parameter set 64
%Scsi% parameter set 55
%SNMP% parameter set 47
%SPX% parameter set 52
%Syslog% parameter set 48
%TCP% parameter set 49
%TokenRingPhysical% parameter set 54
%UDP% parameter set 49

Type 0 CIDFonts 77, 125
Type 1 fonts 77
Type 2 CIDFonts 78, 79, 125
Type 2 fonts 75, 77
Type 14 fonts 75, 77
Type 42 fonts 78

UDP (User Datagram Protocol) 22, 40, 45, 49
%UDP% parameter set 45
(time) 49
UEL command (PJL) 32
unacknowledged messages 42
**undefined** error 3, 94, 100, 103, 107, 108, 109, 110, 111, 122
unencapsulated job 17, 73
unit (definition) 13
UNIX operating system 34, 35, 45
BSD 48
lpr command 38, 39
user parameters 14
See also individual parameters by name
**userdict** dictionary 87
**userdiskpercent** compatibility operator 122

**V**

**ValidNV** system parameter 18
**ValuesPerColorComponent** entry (**DeviceRenderingInfo**
dictionary) 7
**version** value (**systemdict** dictionary) 137
virtual memory (VM) 17, 60, 76, 78
permanent 17
VM. See virtual memory
**VMError** error 17
**VMReclaim** user parameter 14
**VMThreshold** user parameter 14
“vmtx” table (**TrueType** font) 79, 80, 81
%%VMusage: comment (**DSC**) 77

**W**

**WaitTimeout**
  system parameter 18, 97, 113
  user parameter 14, 18, 123
**waittimeout** compatibility value 123
width (glyph) 80, 81
Windows® operating system
drivers 78
**WMode** entry (**CIDFont** dictionary) 80
World Wide Web 74
**Writeable** device parameter 62, 65, 66, 99
writing mode (**CIDFont**) 80

**X**

XNS (Xerox Network System) 22
XON/XOFF flow control 34–35, 112, 119

**Y**

**Year** device parameter 60

**Z**

**Zone** device parameter 44

11x17 compatibility operator 123
11x17tray compatibility operator 124