PostScript Language Reference Manual Supplement

Adobe Systems Incorporated

For Version 2012

November 25, 1992

Adobe Systems Incorporated

Corporate Headquarters
1585 Charleston Road
Mountain View, CA 94043
(415) 961-4400 Main Number

Eastern Regional Office
24 New England Executive Park
Burlington, MA 01803
(617) 273-2120

Adobe Systems Europe B.V.
Europlaza
Hoogoorddreef 54a
1101 BE Amsterdam Z-O
TheNetherlands
+31-20-6511-200

Adobe Systems Japan
Swiss Bank House 7F
4-1-8 Toranomon, Minato-ku
Tokyo 105, Japan
+81-3-3437-8950

PN LPS5211
Contents

List of Figures  v

List of Tables vii

Chapter 1: Overview: PostScript Language Parameters 1

1.1 Introduction  1
1.2 Terminology Used in this Manual  2
1.3 Related Publications  3

Chapter 2: Page Device Parameters 5

2.1 Details Dictionaries  13
2.2 PostScript Language Interface to Fax  14
2.3 Envelope Orientation in User Space  30
2.4 Errors Generated by Page Device Parameters  31

Chapter 3: Interpreter Parameters 35

3.1 Two Kinds of Unencapsulated Jobs  35
3.2 Passwords for System and Device Parameters  36
3.3 User Parameters  36
3.4 System Parameters  38
3.5 Device Parameters  47
3.6 The Fax Environment Interface  80

Chapter 4: Resources 91

4.1 Regular Resources  91
4.2 Implicit Resources  92
4.3 Resources Used to Define New Resources  93

Chapter 5: Other Extensions to PostScript Level 2 95

5.1 Type 6 Halftone Dictionary  95

Chapter 6: Compatibility 97

6.1 Compatibility Operators  98
6.2 Compatibility Operator Descriptions  99
List of Figures

Figure 2.1  Default user space orientation for a portrait envelope PageSize  31
Figure 3.1  Relationship between the communications parameter sets  50
Figure 3.2  Communications parameters sets using NV values  51
Figure 3.3  Communications parameter sets using “hard wired” values  52
List of Tables

Table 2.1  Page device parameters  6
Table 2.2  PageSize entry in the Policies dictionary  12
Table 2.3  Fax options dictionary entries  19
Table 2.4  Entries in a dictionary passed to custom fax job procedures  25
Table 3.1  User parameters  37
Table 3.2  System parameters  39
Table 3.3  Parameters common to all device parameter sets  53
Table 3.4  Parameters common to device sets of type /FileSystem, or

/Communications  53
Table 3.5  Parameters typically present in parameter sets of type /Communications  54
Table 3.6  Parameters typically present in serial communications parameter sets  55
Table 3.7  Parameters typically present in LocalTalk communications parameter sets  59
Table 3.8  Parameters typically present in EtherTalk communications parameter sets  61
Table 3.9  Parameters typically present in parallel communications parameter sets  63
Table 3.10  Parameters typically present in SCSI communications parameter sets  64
Table 3.11  Parameters typically present in disk (/FileSystem) devices  65
Table 3.12  Parameters typically present in cartridge file system (/FileSystem)
devices  69
Table 3.13  Parameters typically present in %os% device parameter sets  72
Table 3.14  Parameters typically present in %Scsi% parameter sets  73
Table 3.15  Parameters typically present in %Engine% device parameter sets  75
Table 3.16  Parameters typically present for the LaserJetIII emulator  77
Table 3.17  Parameters typically present for the LaserJetIIIP emulator  78
Table 3.18  Parameters typically present for the color version of the HP7475A plotter
emulator  79
Table 3.19  Parameters typically present for the Diablo630 emulator  80
Table 3.20  Parameters typically present in the %Fax% device parameter set  81
Table 3.21  Parameters typically present in the %Calendar% device parameter set  86
Table 3.22  Operators for accessing fax job logs  87
Table 3.23  Entries in a job dictionary  88
Table 3.24  Entries in a Calls dictionary  89
Table 4.1  Regular resources  91
Table 4.2  Resources whose instances are implicit  92
Table 4.3  Resources used in defining new resource categories  93
Table 4.4  Description of keys present in an instance of the category OutputDevice  94
Table 5.1  Entries in a type 6 halftone dictionary  96
Table 6.1  statusdict compatibility operators  101
Table 6.2  Stop bits  111
Table 6.3  Data bits  112
Table 6.4  Flow control  112
Table 6.5  Parity  112
Table 6.6  Options byte to devparams conversion  113
Table 6.7  devparams to options byte conversion  113
Table 6.8  SCC compatibility operators  114
Table 6.9  Paper size compatibility operators (in userdict)  115
Table 6.10  Paper tray compatibility operators  116
Table 6.11  Page duplex compatibility operators  117
Table 6.12  Device compatibility operators  119
CHAPTER 1

Overview: PostScript Language Parameters

1.1 Introduction

The purpose of the PostScript Language Supplement for Version 2012 is to provide a supplement to the PostScript Language Reference Manual, Second Edition of standard page device, user, system, and device parameters. This supplement describes new parameters that have been developed since the publication of the manual. It also lists parameters to which changes have been made.

The intended audience for this supplement is the independent software vendors (ISVs) who want to write PostScript language device drivers that can be used for more than one type of device. This catalog will help ISVs produce drivers that support all of the features and capabilities of existing and future PostScript devices.

PostScript Level 2 introduces several operators that take dictionaries as arguments and return dictionaries as results. The key-value pairs in these dictionaries are referred to as parameters because their values typically select optional features or control the operation of some part of the PostScript implementation. The use of dictionaries as containers for parameters provides an extensible method of adding support for new features by adding a new parameter key to the appropriate dictionary. This approach avoids adding new operators to the language on a per-feature basis, thereby maintaining the device independence of the PostScript language.

Specific PostScript implementations include only the parameters that pertain to that product. It is not intended that all of the parameters described in this supplement be present in all products. Once a parameter is defined in any product, it is always used for the same feature in any subsequent products that support it.
1.1.1 Classes of PostScript Level 2 Parameters

Four classes of parameters exist in PostScript Level 2: page device, user, system, and device parameters. Each class corresponds to a pair of PostScript language operators: one that returns the current values of a set of parameters and one that takes as an argument a collection of parameters that are to be set. These operators are:

- `currentpagedevice` 
- `setpagedevice`
- `currentuserparams` 
- `setuserparams`
- `currentsystemparams` 
- `setsystemparams`
- `currentdevparams` 
- `setdevparams`

In terms of functionality, parameters fall into two broad categories. The first category corresponds to printing capabilities (optional trays, duplex, collating, etc.). These are the page device parameters; they are described in Chapter 2. The second category corresponds to the operation and behavior of the PostScript interpreter. These are the interpreter parameters, which include the system, user, and device parameters. These parameters are described in Chapter 3.

1.1.2 PostScript Level 2 Resources

In Level 2, PostScript objects such as fonts, patterns, filters, and so on, can be managed as open-ended collections of resources. The resources common to all implementations of PostScript language version 2012 are listed in Chapter 4.

1.1.3 Other Extensions to PostScript Level 2

The Postscript Language continues to evolve. Chapter 5 lists those extensions to the language for version 2012.

1.1.4 Compatibility Operators

For compatibility with existing Level 1 PostScript language driver software, which might depend on operators that were often present in PostScript Level 1 products, a collection of compatibility operators and procedures is included in each Level 2 implementation. These compatibility operators are described in Chapter 6.

1.2 Terminology Used in this Manual

Throughout this manual, the following terms are used:

- `device`: A device is defined as a piece of hardware under the control of a PostScript interpreter. There are several categories of devices:
page device: A page device can be, for example, a laser print engine producing paper output.

communication device: A communication device can be, for example, serial, parallel, or LocalTalk communications hardware and software.

filesystem device: A filesystem device can be, for example, a disk or cartridge system.

host: A host is defined as a computer system (for example, a personal computer or workstation) connected to a PostScript printer product via one of its communication devices. The host sends PostScript language programs over the communication channel to the printer. The printer executes them.

PostScript interpreter: A PostScript interpreter is defined as a body of software that executes programs written in the PostScript language and produces effects such as generating printed output on a page device.

PostScript product: A PostScript product is defined as a system consisting of a PostScript interpreter controlling one or more devices.

1.3 Related Publications

Adobe Communications Protocols Specification, available from the Adobe Developers Association, describes several protocols that can be used to communicate over a serial or parallel connection to a PostScript printing device.

PostScript Language Program Design (Reading, MA: Addison-Wesley, 1988) teaches programming principles unique to the Level 1 PostScript language and contains many usable samples. It is for programmers interested in the effective and efficient design of PostScript language programs and printer drivers.

This chapter lists the page device parameters. For more information about how the `setpagedevice` operator is used to set up a raster output device, refer to section 4.11, “Device Setup” in the *PostScript Language Reference Manual, Second Edition*.

Two operators, `currentpagedevice` and `setpagedevice`, respectively read and set the parameter values.

The following page device parameters are described in the *PostScript Language Reference Manual, Second Edition*. The description of these parameters is unchanged.

<table>
<thead>
<tr>
<th>AdvanceDistance</th>
<th>AdvanceMedia</th>
<th>BeginPage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collate</td>
<td>CutMedia</td>
<td>Duplex</td>
</tr>
<tr>
<td>EndPage</td>
<td>HWRResolution</td>
<td>ImagingBBox</td>
</tr>
<tr>
<td>Install</td>
<td>Jog</td>
<td>ManualFeed</td>
</tr>
<tr>
<td>MediaColor</td>
<td>MediaType</td>
<td>MediaWeight</td>
</tr>
<tr>
<td>MirrorPrint</td>
<td>NegativePrint</td>
<td>NumCopies</td>
</tr>
<tr>
<td>Orientation</td>
<td>OutputAttributes</td>
<td>OutputFaceUp</td>
</tr>
<tr>
<td>OutputType</td>
<td>PageSize</td>
<td>Policies</td>
</tr>
<tr>
<td>Separations</td>
<td>Tumble</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1 describes the page device parameters that have been defined or amended since the publication of the *PostScript Language Reference Manual, Second Edition*. 
### Table 2.1 Page device parameters

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bind</td>
<td>integer</td>
<td>This parameter requests that the document be bound. The job will be bound at a specific time indicated by an integer code:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Code</strong></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>Booklet</td>
<td>boolean</td>
<td>This parameter requests that the document be stapled, trimmed, and folded into booklet form.</td>
</tr>
<tr>
<td>BookletDetails</td>
<td>dictionary</td>
<td>This dictionary describes product-specific details related to how a document is to be bound. For more information on Details dictionaries, see section 2.1, “Details Dictionaries.”</td>
</tr>
<tr>
<td>DeviceRenderingInfo</td>
<td>dictionary</td>
<td>This dictionary provides a location for individual OEMs or products to specify their own device rendering parameters. The only pre-specified key is Type of type integer.</td>
</tr>
<tr>
<td>ExitJamRecovery</td>
<td>boolean</td>
<td>If true, pages that jam in the exit path are reprinted. If false (jam recovery disabled), performance might be improved because more overlapping of page processing is possible.</td>
</tr>
<tr>
<td>FaxOptions</td>
<td>dictionary</td>
<td>For a description of the FaxOptions key as well as the fax interface, see “PostScript Language Interface to Fax,” section 2.2.</td>
</tr>
<tr>
<td>Fold</td>
<td>integer</td>
<td>This parameter requests that the document be folded. The job will be folded at a specific time indicated by an integer code:</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not fold.</td>
</tr>
<tr>
<td>1</td>
<td>Fold at device deactivation.</td>
</tr>
<tr>
<td>2</td>
<td>Fold at the end of the job.</td>
</tr>
<tr>
<td>3</td>
<td>Fold after each set.</td>
</tr>
<tr>
<td>4</td>
<td>Fold after each <code>showpage</code> or <code>copypage</code>.</td>
</tr>
</tbody>
</table>


**FoldDetails** dictionary

This dictionary describes product-specific details related to how a document is to be folded. For more information on **Details** dictionaries, see section 2.1, “Details Dictionaries.”

**InputAttributes** dictionary

If **InputAttributes** is null instead of a dictionary, the PostScript interpreter has no previous knowledge of the available media. When **setpagedevice** is executed, the interpreter simply presents media selection requests to the device implementation, which is fully responsible for determining if they can be satisfied. This arrangement exists in products where actual printing of the output is deferred to some process not directly under the control of the PostScript interpreter.

The **InputAttributes** dictionary normally contains an entry for each input media source. The entry consists of an integer representing the input paper tray and an associated dictionary. A new key may be found on some printers within a tray dictionary. The new key is called **InsertSheet** and is a boolean indicating whether or not the tray is a special insert sheet tray. This new media key is used during the media matching process and compared against the new **setpagedevice** key **InsertSheet** (described next).

**InsertSheet** boolean

This parameter specifies whether or not to select inserted media. **setpagedevice** compares it with the **InsertSheet** value, if any, in the **InputAttributes** entries for all media that it considers. Refer to section 4.11.4, “Media Selection” in the *PostScript Language Reference Manual, Second Edition*, keeping in mind that **InsertSheet** is also an input media entry found within the **InputAttributes** dictionary.

A side-effect of executing **setpagedevice** with **InsertSheet** equal to **true** and selecting an insert sheet tray is that the imageable area gets set to a zero-area region to ensure that nothing is imaged on the inserted sheet. That is, the insert sheet is explicitly not imaged. Photographic material is a good example of **InsertSheet** media which has the special requirement that it cannot tolerate being imaged to, nor sent through the fuser without major damage. This implies that the media pulled from an **InsertSheet** slot does not go through the normal paper path.

Here is an example of how to use **InsertSheet**.
ManualFeedTimeout

integer The number of seconds the printer waits for a page to be fed manually before generating a timeout error. A zero value means no timeout (infinite wait).

Margins array If the device supports multiple resolutions (that is, different values of HWResolution), the margin values are interpreted according to some canonical default resolution and are scaled appropriately at other resolutions. This ensures they represent the same physical distance when the resolution is varied. The canonical default resolution is product dependent and specified in the Addendum for the product. For more information on Margins, see Table 4.11 in section 4.11.3 of the PostScript Language Reference Manual, Second Edition.

OutputDevice name This parameter selects an output device in environments in which the PostScript interpreter can generate output for multiple page devices. In some environments, it selects among different types of output devices, such as a printer and a fax modem, a printer and a display screen, or a printer and an imagesetter. In other environments, it may select among similar devices, such as two or more imagesetters.

When the value of OutputDevice changes, the usual inheritance of values not specified in the operand to setpagedevice does not happen. Instead, all new values are generated in a manner that is specific to each product. Also, the set of acceptable keys for setpagedevice can change when changing the value of OutputDevice, since different devices have different features that can be controlled or queried.

OutputPage boolean If true, processing is normal. If false, no pages are actually printed, but all other processing is done as if the page were to be printed, including rasterizing to a frame buffer. Thus when OutputPage is false, the time to process a page includes everything except time spent waiting for the marking engine.

Furthermore, rasterization occurs synchronously with execution of showpage instead of being overlapped with execution of subsequent pages. This facilitates measuring the complete cost of page execution.
PostRenderingEnhance
   boolean If true, product-specific image enhancements are enabled. These enhancements are made after the page is rasterized in memory.

PostRenderingEnhanceDetails
dictionary This dictionary describes product-specific details related to the post-rendering image enhancement. For more information on Details dictionaries, see section 2.1, “Details Dictionaries.”

PreRenderingEnhance
   boolean If true, product-specific image enhancements are enabled. These enhancements are made before the image is rasterized in memory.

PreRenderingEnhanceDetails
dictionary This dictionary describes product-specific details related to the pre-rendering image enhancement. For more information on details dictionaries, see section 2.1, “Details Dictionaries.”

ProcessColorModel
   name or string This name or string value specifies the colorant model used for rendering process colors in the device. It affects rendering for all color spaces, with the exception of Separation color spaces that actually produce separations. It does not affect the interpretation of color values in any color space; it controls only the rendering method.

Legal values are DeviceGray, DeviceRGB, DeviceCMYK, DeviceCMY, and DeviceRGBK. For example, DeviceRGB specifies that the process colorants are named Red, Green, and Blue; DeviceCMYK specifies Cyan, Magenta, Yellow, and Black. These are the process colorant names used to select half-tones in a type 5 halftone dictionary and to control the production of separations in SeparationColorNames and SeparationOrder.

Each of the ProcessColorModel values implies a specific native color space for the device. The native color space is the PostScript language device color space into which user-specified colors are converted if necessary; see section 6.2 in the PostScript Language Reference Manual, Second Edition.

- DeviceGray, DeviceRGB, and DeviceCMYK select the correspondingly named native device color space.

- DeviceCMY and DeviceRGBK both select DeviceRGB as the native device color space, but they cause the device to render the DeviceRGB color values in special ways. For DeviceCMY, the device renders the RGB colors using the complementary subtractive colors. For DeviceRGBK, the device uses a separate rendering method for RGB color values that represent pure shades of gray.
SeparationColorNames

array
This parameter specifies those Separation color spaces that the device supports. This array can contain either names or strings, e.g., 
[/ Pink /Green] or [(Pink) (Green)] or [/ Pink (Green)].

If the name used in a [/Separation name ... ] setcolorspace operation is included in this array, that colorant will be used, rather than the alternate color space. Any other color will be mapped to one or more of the named colors through the alternate color space and tintTransform procedure of the setcolorspace operator. This is described in section 4.8.4 of the PostScript Language Reference Manual, Second Edition.

The names of the colorants of the native color space are included implicitly, regardless of the contents of the array. Thus:

- for DeviceCMY, the empty array [] is equivalent to [/Cyan /Magenta /Yellow].
- for DeviceCMYK, the empty array [] is equivalent to [/Cyan /Magenta /Yellow /Black].
- for DeviceRGB, the empty array [] is equivalent to [/Red /Green /Blue].
- for DeviceRGBK, the empty array [] is equivalent to [/Red /Green /Blue /Black].
- for DeviceGray, the empty array [] is equivalent to [/Gray].

SeparationOrder

array
If separations are being made, this parameter specifies that they be produced in the order given by the array of color names (where an array can contain either names or strings, such as [/Cyan /Magenta] or [(Cyan) (Magenta)]). Legal values are the names of the colorants of the native color space, as well as any additional names specified by SeparationColorNames.

A separation will be produced for each occurrence of a name; multiple occurrences will produce multiple separations. No separations will be produced for colors whose names are not given regardless of their appearance in SeparationColorNames. The named separation color space is defined (as opposed to reverting to the alternative color space), but the output for that separation is discarded when a certain color name is not given.

An empty array [] requests that separations for all colors of the native color space, as well as all colors requested by SeparationColorNames, be produced in an unspecified order.
When not making separations, some devices may use `SeparationOrder` to determine the colorants and the order in which they are to be applied to the composite image.

**Signature** boolean If *true*, the job will be “signatured.” That is, pages of a document will be arranged so that, when folded, the pages will be in the right order. How signaturing is performed is device-dependent. On some devices, the engine may provide the resources (memory, disk space) to signature the job. On other devices, the interpreter may have to reorder the virtual pages in order to deliver the pages to the engine in the correct order. In the latter case, a `Signature` value of *true* implies that the interpreter must store the results of executing the page description for multiple pages in order to deliver the pages correctly ordered. This use of `Signature` is supported by relatively few products and is subject to resource limits in products that do support it.

**SlipSheet** integer This parameter requests that slip sheets (slip sheet media selection is product specific) be inserted. There is no way to render a slip sheet; the engine is simply told when to insert them. For example, a slip sheet can be a colored sheet of paper that visually separates multiple copies or a tabbed sheet that separates chapters. Compare with the description of `InsertSheet` above.

Slip sheets will be inserted at specific times indicated by an integer code:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not insert slip sheets.</td>
</tr>
<tr>
<td>1</td>
<td>Insert slip sheet at device deactivation.</td>
</tr>
<tr>
<td>2</td>
<td>Insert slip sheet at the end of the job.</td>
</tr>
<tr>
<td>3</td>
<td>Insert slip sheet at the end of the set.</td>
</tr>
<tr>
<td>4</td>
<td>Insert slip sheet after each <code>showpage</code> or <code>copypage</code>.</td>
</tr>
</tbody>
</table>


**Staple** integer This parameter requests that the job be stapled. The job will be stapled at a specific time indicated by an integer code:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not staple.</td>
</tr>
<tr>
<td>1</td>
<td>Staple at device deactivation.</td>
</tr>
<tr>
<td>2</td>
<td>Staple at the end of the job.</td>
</tr>
<tr>
<td>3</td>
<td>Staple after each set.</td>
</tr>
<tr>
<td>4</td>
<td>Staple after each <code>showpage</code> or <code>copypage</code>.</td>
</tr>
</tbody>
</table>

StapleDetails dictionary This dictionary describes product-specific details related to how a document is to be stapled. For more information on Details dictionaries, see section 2.1, “Details Dictionaries.”

TraySwitch boolean If true, automatic tray switching is provided. This option is offered by some devices with multiple input trays. When one input tray runs out of media, another tray with the same type of media can be used automatically, without alerting you that the printer is out of media.

Trim integer This parameter requests that the job be trimmed. The job will be trimmed at a specific time indicated by an integer code:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not trim.</td>
</tr>
<tr>
<td>1</td>
<td>Trim at device deactivation.</td>
</tr>
<tr>
<td>2</td>
<td>Trim at the end of the job.</td>
</tr>
<tr>
<td>3</td>
<td>Trim after each set.</td>
</tr>
<tr>
<td>4</td>
<td>Trim after each showpage or copypage.</td>
</tr>
</tbody>
</table>


Table 2.2 describes changes to the PageSize entry in the Policies page device dictionary.

**Table 2.2** PageSize entry in the Policies dictionary

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>PageSize</td>
<td>integer</td>
<td>This parameter specifies the recovery policy to use when the PageSize cannot be matched (within a tolerance of 5 units) with any available media. The following policy value has been added to Table 4.14 in section 4.11.5, “Policies,” in the PostScript Language Reference Manual, Second Edition:</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Disable media selection altogether and impose the requested PageSize on the previously selected medium, without adjusting it in any way. That is, set up the page device as if the selected medium were of the requested size, ignoring the medium's actual size. The manner in which the page image will be positioned on the medium is product dependent and unpredictable.</td>
</tr>
</tbody>
</table>

When the PageSize policy is 7, it takes effect during every execution of setpagedevice. This is unlike all other policies, which take effect only if a request cannot be satisfied.
This policy exists solely for use in the emulations of certain Level 1 compatibility operators that perform media selection and page device setup separately. **PageSize** policy 7 should never be used in a Level 2 application. Its semantics violate the Level 2 page device model, and documents using it are not portable.

### 2.1 Details Dictionaries

Certain page device features have many variables which determine precisely how the feature functions; these variables may be quite different on different products. Such a feature is enabled or disabled by a primary page device entry while the exact way in which the feature functions is determined by secondary entries in a **Details** dictionary page device entry. This allows an application that is not knowledgeable about the details of the feature to enable and disable it, while more sophisticated utilities can be used to configure the details separately.

An example of this is the stapling feature. Many applications will want to either enable or disable stapling with the assumption that the number, location, and orientation of the staples has been configured correctly. The nature of the configuration will be dependent on the printing device. For example, for some engines it may be possible to specify an arbitrary staple location on the sheet while on others, staples may be placed only in the four corners.

Primary page device entries for such features are either booleans or integers. If the value is a boolean, then the feature is enabled if the value is true and disabled if the value is false. If the value is an integer, the feature is disabled if the value is zero. The non-zero values enable the feature in different ways that are consistent across all products. For example, the binding feature can be enabled for binding at the end of device deactivation, at the end of a job, or at the end of each set.

A consistent naming convention is used for **Details** dictionaries. The name of the dictionary is the name of the primary key with “**Details**” appended. For example, if the **Staple** feature is present and has a details dictionary, this dictionary is named **StapleDetails**.

A **Details** dictionary will be present for a given feature on a given product only if additional information beyond that of the primary entry is needed to control it. For example, a product supporting a post rendering enhancement feature which can only be enabled or disabled with no further control will not have a **Details** dictionary for this feature. Such a **Details** dictionary would be present on a printer with more configurable post rendering enhancement. Applications which are simply enabling and disabling a feature should never reference a **Details** dictionary. More sophisticated applica-
tions wishing to control a **Details** dictionary should never assume that one is present unless they know the exact nature of the printing device on which they are executing.

During the execution of **setpagedevice**, the entries in any **Details** dictionary must always be syntactically correct, but the validity of the values is only checked if the feature will be enabled for the page device in effect as a result of **setpagedevice**. As with all page device entries, syntactically incorrect settings result in appropriate PostScript language errors (for example, **typecheck**) and invalid values result in policy being consulted.

### The Type Entry

Every **Details** dictionary has a **Type** entry whose integer value completely determines how the **Details** dictionary entries effect the feature. That is, if two different products have **Details** dictionaries for the same feature and the **Type** entry is the same for each, then the dictionaries will have exactly the same named entries and the syntax and semantics of each entry will be the same. This allows an application, based solely on the value of the **Type** entry, to change entries in a **Details** dictionary for a feature.

If **Details** dictionary entries are being set, whether the new dictionary overwrites the current one or is merged with it, is determined by the **Type** entry. The criteria for merging versus overwriting is product dependent. **Details** dictionaries and their associated **Type** entries are registered by Adobe Systems.

### 2.2 PostScript Language Interface to Fax

This section is intended for use by programmers who are building PostScript language drivers and utility software to work with PostScript fax printers. PostScript language drivers create pages to be printed and faxed, and thus need to know how to create PostScript language fax jobs in addition to PostScript language print jobs.

PostScript fax printers require utility software on the host computer to set variables in the printer, such as time-of-day, number of rings to answer on, speaker settings, etc. These settings persist across multiple fax jobs and multiple users. Programmers developing utility software need a language interface to read and write this data.

The description of the PostScript language interface to fax is broken down into two major sections.

- “PostScript Language Interface to Fax,” section 2.2, describes the interface which controls individual fax jobs. “Some Sample Fax Jobs,” section 2.2.7 presents several example raster and PostScript language fax jobs.
• “The Fax Environment Interface,” section 3.6, describes the device parameters and administrative resources needed to set, control, and examine the fax environment shared by all jobs and all users connected to the fax printer.

2.2.1 The Fax Job Interface

PostScript fax printers may send faxes in two different ways: they may send raster images, and they may send PostScript language files. Raster images may be received by any CCITT Group 3 fax machine. PostScript language files may only be received by suitably equipped machines. The PostScript interfaces for these two different, but related, methods are described in sections 2.2.2, “Sending Raster Images” and 2.2.3, “Transmitting PostScript Language Files.” Common to both is a collection of dictionary keys which contain information, such as telephone numbers, necessary to carry out the fax operation. These keys are described in section 2.2.4, “The Fax Options Dictionary Keys.”

Cover sheets, transmission reports and page captions are traditional elements of fax communication. The job interface makes it possible to generate these items. This is described in section 2.2.5, “CoverSheet, Confirmation, and PageCaption Procedures.”

Fax Job Transmissions

Fax job transmissions can be made to single or multiple destinations.

A fax transmission in raster form to a single destination will be made in a single call if there is sufficient storage to hold all of the raster pages of the job. However, the transmission may be broken into multiple calls if there is not sufficient storage. If the storage for the raster pages is exhausted, then a call will be made before rasterization is complete. Transmission and rasterization will then take place simultaneously. If transmission succeeds in sending all prepared pages before rasterization of the job is finished, the call will end. A subsequent call will be made either when all of the rasterization is done or when storage is again exhausted. Thus, a transmission may be broken into multiple calls. It is also possible that when transmission and rasterization are taking place simultaneously, rasterization keeps up with transmission. In this case, this transmission will be the last call for the job.

If the storage in use is disk, there will typically be sufficient storage for all of the raster pages of a job and the fax will be transmitted in a single call.

It is possible to request that a transmission take place at a particular time. These delayed transmissions, of course, are made with a single call. Since the pages of a raster job are prepared when the job is submitted, all of the pages of a delayed raster transmission must fit in storage otherwise the job will fail.
The transmission of a PostScript language file to a single destination is always made in a single call. This call takes place only after the entire file is in storage. If the file will not all fit in storage, the job will fail.

The transmission of raster files and PostScript language files to multiple destinations is described in “Broadcast Transmission of Faxes,” section 2.2.6.

### 2.2.2 Sending Raster Images

PostScript language files can be sent as raster images to CCITT Group 3 fax machines by selecting `/Fax` as the current page device with the `setpagedevice` operator:

```latex
<< /OutputDevice /Fax
other key-value pairs >> setpagedevice
```

With a call to `setpagedevice`, one of three cases is possible:

1. A new page device is established that does not change the value of `OutputDevice` in the current page device. In this case, entries in the new page device which are not specified in the argument dictionary to `setpagedevice` are inherited from the current page device.

2. A new page device is established when there is no current page device. In this case, entries in the new page device which are not specified in the argument dictionary to `setpagedevice` are initialized to product-dependent default values.

3. A new page device is established that changes the value of `OutputDevice` in the current page device. In this case, entries in the new page device are not inherited from the current page device, rather the entries in the new page device which are not specified in the argument dictionary to `setpagedevice` are initialized to product dependent default values.

When the `OutputDevice` is `/Fax`, one of the entries in the page device is the `FaxOptions` dictionary. Values in the `FaxOptions` dictionary, if any, supplied as part of the argument dictionary to `setpagedevice` are merged, to one level, into the `FaxOptions` dictionary held in the page device. This is similar to the treatment of `InputAttributes` and `OutputAttributes`. Section 2.2.4, “The Fax Options Dictionary Keys,” describes the entries found in the `FaxOptions` dictionary and also lists typical default values.

If the call to the `setpagedevice` operator does not change the `OutputDevice`, (case 1 above) then the `FaxOptions` dictionary before and after the merge of the argument dictionary is considered. If the `FaxOptions` dictionary has been changed in any way, then it is assumed that this is a new transmission; the previous one is finished (coversheets are generated, the
phone call is queued, etc.) and a new transmission context is started. The reliable technique is to establish all the FaxOptions during the first call to setpagedevice that has the OutputDevice set to /Fax.

The FaxOptions DialCallee Key

One of the entries in the FaxOptions dictionary, the DialCallee key, contains the phone number for the fax machine to call. If the DialCallee key has a null value, the fax transmission cannot proceed, and a configurationerror will be raised. The DialCallee key is described in greater detail in section 2.2.4.

If entries in the new page device are initialized to product-dependent default values (see cases 2 and 3 above), then the argument dictionary must contain an entry for FaxOptions which in turn must contain a value for the DialCallee key. If the FaxOptions dictionary does not contain a value for the DialCallee, it will be assigned the product-dependent default value (a null). Fax transmission cannot proceed using a null value as the phone number to call and a configurationerror will be raised.

2.2.3 Transmitting PostScript Language Files

It is possible to transmit jobs as PostScript language files between consenting PostScript fax printers. Two entries in the argument dictionary passed to the faxsendps operator control this transmission. These are RevertToRaster and PostScriptPassword. The %Fax% device’s ReceivePostScript and PostScriptPassword parameters (see Table 3.20 on page 81) control the receipt of PostScript language jobs by a fax printer.

A job may be sent in PostScript language form by using the faxsendps operator found in the FaxOps ProcSet:

```
file OptionsDict
/FaxOps /ProcSet findresource /faxsendps get exec
```

The file object is customarily the object returned by currentfile, representing the contents of the remainder of the current job. The faxsendps operator will either send this file to the remote printer to be executed as a PostScript language job, or execute it locally producing a raster fax. The following four steps describe how this is done.

1. The file is read to EOF with the contents of the file being saved internally. If there is sufficient storage to hold the entire file, proceed to step 2, below.

If there is not sufficient storage to hold the entire file, RevertToRaster is consulted. If RevertToRaster is false, the job fails and the message “Storage to assemble PostScript transmission exhausted” will be sent to stdout. If it is true, then the local machine will execute the equivalent of
resulting eventually in a raster image fax transmission. The process terminates here, and steps 2 through 4 do not occur.

2. The DialCallee phone number in the OptionsDict is dialed, and the receiver queried regarding its willingness to receive PostScript language files from the sender. The PostScriptPassword from the OptionsDict may be used in this process.

3. If the receiver is willing, the contents of the file saved internally will be sent to the remote machine.

4. If the receiver is not willing, the sending machine hangs up. RevertToRaster is consulted. If it is true then the local machine will execute the equivalent of

   ```<</OutputDevice /Fax /FaxOptions OptionsDict>>
   setpagedevice
   file cvx exec```

   resulting eventually in a raster image fax transmission. If RevertToRaster is false, no transmission will occur. The job will end and the message “Callee rejected PostScript transmission” will appear in the log.

2.2.4 The Fax Options Dictionary Keys

Both the FaxOptions dictionary and the dictionary passed to the faxsendps operator contain the information used to direct a fax transmission. These dictionaries also provide a number of ways of customizing fax jobs. Information in the dictionaries is placed on cover sheets and transmission reports, and is recorded in logs. Procedures in the dictionaries are used for page captions, cover sheets, and transmission reports. Other values in the dictionaries control broadcast, delayed transmission, and high resolution capabilities.

Table 2.3 lists all of the keys present in these dictionaries. Only these keys may be present. For each key, the allowed type of the associated value and typical default values are listed.

As described above, if there is no current page device or the new page device’s OutputDevice is different from that of the current page device, the FaxOptions dictionary initially contains the keys listed below with their associated default values. Merging of the setpagedevice’s argument dictionary then takes place. For the faxsendps operator, keys not present in its argument dictionary are treated as if their default values were given.
Table 2.3  Fax options dictionary entries

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalleePhone</td>
<td>string or null</td>
<td>This key indicates the human-readable telephone number of the callee fax machine. The value of CalleePhone is used for the Confirmation, Cover-Sheet, and PageCaption procedures. It differs from DialCallee primarily in that it omits or alters routing prefixes and suffixes. For example, the Swiss number in the example under DialCallee might be written here (0041-5-55-55-55732) or (CH 5-55-55-55732). If the value of CalleePhone is null, the value of DialCallee is used.</td>
</tr>
<tr>
<td>CallerID</td>
<td>string or null</td>
<td>The ID is defined by the CCITT (1988) fax protocol and is a string of up to 20 characters which the caller uses to identify himself to the callee. If it has a default null value, then the value of ID from the %Fax% device parameter set will be used (see Table 3.20 on page 81). If ID is not set, then the string returned by the system parameter PrinterName will be used. If this string is greater than 20 characters, then the left-most 20 characters will be used.</td>
</tr>
<tr>
<td>CallerPhone</td>
<td>string or null</td>
<td>This key indicates the human-readable telephone number of the caller fax machine.</td>
</tr>
<tr>
<td>Confirmation</td>
<td>procedure or null</td>
<td>Prints a confirmation sheet (on the local sending print mechanism) for this job. The Confirmation procedure is executed when the fax job has finished and the transmission, if any, has completed. You can omit confirmation reports (by setting Confirmation to null) or customize them (by providing your own procedure). You may, for example, want to print a report only if there is an unusual printer status. The Confirmation procedure must explicitly call the showpage operator to print the page. This allows confirmation to be sent to the host computer instead of being printed. Therefore, if an alternate definition of the confirmation report is to print upon job completion, the procedure definition must include a call to showpage. The default value depends on the value of the %Fax% parameter DefaultConfirmOn. If DefaultConfirmOn is true, (see Table 3.20 on page 81) then the default value is{/DefaultConfirmation faxopsexec} which provides a simple, generic report. If DefaultConfirmOn is false, then the default value of Confirmation is null.</td>
</tr>
</tbody>
</table>
**Copies**
array of dictionaries or null

This array of dictionaries provides a means by which the same raster or PostScript language file may be broadcast to multiple recipients. The only keys allowed in these dictionaries are the same ones allowed in the FaxOptions dictionaries (except that Copies may not be present). Each dictionary from the array is merged with the original FaxOptions dictionary (without the Copies entry) and used to direct the “mailing” of the common body of the fax to a different recipient. Where the same key is defined in both dictionaries, the value from the Copies element takes precedence. For more information on this broadcast facility and a discussion of some of the details and exceptions to this dictionary merging, see section 2.2.6, “Broadcast Transmission of Faxes.”

**CoverNote**
array of strings or null

This key passes information to the CoverSheet procedure. This information is intended to be a quick note on the cover sheet. It could also be used for the entire message if the fax consists of only a cover sheet.

**CoverSheet**
procedure or null

This PostScript language procedure generates cover sheets for the document. Fax documents are often preceded (and occasionally followed) by cover sheets. These serve some of the same purposes as envelopes for normal mail. They specify to whom the document is to be delivered, how he might be found, who claims to be the sender, what to do if the document is found to be incomplete, and so on. You can define CoverSheet as a PostScript language procedure that produces a customized page. The details for writing these procedures are given in section 2.2.5, “CoverSheet, Confirmation, and PageCaption Procedures.”

Sometimes the PostScript language job will contain all of the pages that the sender wants the recipient to receive. For example, a one-page letter that contains an inside address and a return address may not require a cover sheet as well. Whenever CoverSheet is defined as null, the fax software will not generate cover pages.

The default value for this key depends on the %Fax% parameter, DefaultCoversOn, (see Table 3.20 on page 81). If this parameter is true, then the default is

{/DefaultCoverSheet faxopsexec}

which will generate simple cover sheets using Sender..., Recipient..., and other optional values from the FaxOptions dictionary. If the parameter is false, the default value will be null.

The CoverSheet procedure should not include a showpage operator. showpage is called automatically after the CoverSheet procedure is invoked.
CoverSheetOnly  boolean  This key indicates that it is all right to send an “empty” job. If this boolean is `false` and the PostScript language job produces no pages, then no phone call is made and nothing is sent. If this boolean is `true` and `CoverSheet` is non-null, then the page generated by the cover sheet procedure is sent in any case. The default value is `false`.

DialCallee  string  This key indicates the phone number of the fax machine to call. This string will be sent to the telephone auto-dialer in the fax printer. It consists of a sequence of the following characters:

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>begin pulse dialing (initial default)</td>
</tr>
<tr>
<td>T</td>
<td>begin DTMF (Touch-Tone) dialing</td>
</tr>
<tr>
<td>0-9</td>
<td>send signal digit to telephone exchange</td>
</tr>
<tr>
<td>* #</td>
<td>send DTMF symbol to telephone exchange, pause for 2 seconds</td>
</tr>
<tr>
<td>W</td>
<td>wait for dial tone</td>
</tr>
<tr>
<td>others</td>
<td>ignore</td>
</tr>
</tbody>
</table>

For example, to dial a Swiss fax machine from a private branch exchange in the USA, you might use the string:

```
T9,011-41-5-55-55-55732#
```

Notice that this string contains a routing prefix to establish tone, international switching from the PBX (T9,011). It also contains the suffix #, which is used by most American telephone operating companies as a signal that all digits have been sent and switching should commence immediately. This string has a maximum length of 100 characters.

ErrorCorrect  boolean  Controls whether error correction should be attempted in the transmission. The receiving machine must be capable of error correction in order for the attempt to succeed. If the receiver does not have the capability, the transmission will take place without it. The default value is `true`.

FaxType  integer or null  This key determines how the actual page contents are prepared and transmitted. If the value is an integer, it should be 0 or 1 coding the following information:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>use standard CCITT group 3 resolution</td>
</tr>
<tr>
<td>1</td>
<td>use fine CCITT group 3 resolution</td>
</tr>
</tbody>
</table>

If the `FaxType` is 0, then the transmitted y-resolution will be approximately 100 lines per inch; if 1, the y-resolution will be twice as fine (and the transmission time will be correspondingly longer). As the CCITT standards evolve, other integers may trigger other modes. If the value associated with
FaxType is null, then the value of the %Fax% device parameter DefaultResolution, which is 0 or 1, will be used to select the resolution (see Table 3.20).

MailingTime  
array of integers or null  
This key indicates when the fax message is to be transmitted. This is used, for example, to send documents at night when phone rates are lower. The value is an array of integers with entries as follows:

<table>
<thead>
<tr>
<th>Index</th>
<th>Meaning</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Year</td>
<td>[1980-2079]</td>
</tr>
<tr>
<td>1</td>
<td>Month</td>
<td>[1-12]</td>
</tr>
<tr>
<td>2</td>
<td>Day</td>
<td>[1-31]</td>
</tr>
<tr>
<td>3</td>
<td>Hour</td>
<td>[0-23]</td>
</tr>
<tr>
<td>4</td>
<td>Minute</td>
<td>[0-59]</td>
</tr>
<tr>
<td>5</td>
<td>Second</td>
<td>[0-59]</td>
</tr>
</tbody>
</table>

If this key has the default null value associated with it, then the time of job submission will be used, meaning “send immediately.” Delayed mailings are only possible if the machine has sufficient storage (disk or RAM) to save the entire job until it is to be sent.

MaxRetries  
integer or null  
This key indicates how many additional tries after the first should be made before giving up on the transmission of a fax message. Attempts can fail, for example, because of a busy or no-answer when the call is placed. If the value is null, then the value used is given by the %Fax% parameter DefaultRetryCount (see Table 3.20). The maximum value is 100. If an attempt is being made to send a PostScript language file and the job reverts to raster, the count of the number of failed calls, in effect, is reset to 0. This is because reverting is equivalent to submitting a new job.

nPages  
integer or null  
This key supplies the application’s estimate of the number of pages in the job exclusive of automatically generated cover sheets. In some cases the transmitting machine has not processed the complete job before the transmission starts (and the cover sheet must be sent). In this case, the number of pages will not be known to the printer, so if nPages is provided, it will be used. If nPages is null (the default) then the cover sheet will list an unknown number of pages.

PageCaption  
procedure or null  
This PostScript language procedure generates information lines on the top of transmitted fax pages. Typically, such information will contain the sender’s name, the recipient’s name, and the current page number. The PageCaption procedure should not call the showpage operator since the procedure only makes alterations to the existing page image. The generation of captions will not occur if PageCaption is null. For more information on the interface to a PageCaption procedure, see section 2.2.5, “CoverSheet, Confirmation, and PageCaption Procedures.”
The default value for this key depends on the %Fax% parameter, **DefaultCaptionOn** (see Table 3.20). If this parameter is *true*, then the default is the procedure

```
{/DefaultPageCaption faxopsexec}
```

If **DefaultCaptionOn** is *false*, then the default value is null.

**PostScriptPassword**

(string or null) This parameter specifies a password to use in gaining permission from the callee to transmit the fax job as a PostScript language file. This entry is used only by the `faxsendps` operator. The default is null, which means that no password is being supplied. See section 3.6.2, “The %Fax% Device”, for a description of how this password is used.

**ProcInfo**

(dictionary or null) This dictionary may be used to supply any number of additional application-specific key-value pairs. These key-value pairs are used to convey variable information for application-defined cover sheets, confirmation reports and page captions.

**RecipientID**

(string or null) This key contains a string that uniquely identifies the recipient within the organization (at that fax number). It is intended to be computer readable and usable for subsequent electronic delivery of the fax message within the receiver’s organization. If this value is null, a string of nulls will be transmitted. The semantics of this parameter, as well as the protocol for delivering it electronically, are not specified by Adobe Systems.

**RecipientMailStop**

(string or null) This key contains information helpful for hand delivery of the fax message. For example, Mail Stop 23A, Bldg. 19.

**RecipientName**

(string or null) This key indicates the document’s intended recipient; for example, Dr. John Doe. A default value for **RecipientName** leads the code to seek an alternative non-null value to store in the job log. The first alternative is to use the value of **RecipientOrg**. If **RecipientOrg** also has a default value, then the value of **CalleePhone** is used. If **CalleePhone** also has a default value, then the value of **DialCallee** is used.

**RecipientOrg**

(string or null) This key indicates the recipient’s company or organization name. This value is also stored in the job logs. The fall back sequence is **RecipientName** and then **DialCallee**.

**RecipientPhone**

(string or null) This key indicates the recipient’s voice telephone number. It is not the same as **DialCallee**, which is the fax phone number. The **RecipientPhone** is used for purposes like custom cover sheets that would give routing instructions to
an attendant on the fax receiving end. As with RecipientName, if the value associated with this key is the default, fall backs are sought to store in the log. The fall back sequence is CalleePhone and then DialCallee.

Regarding
string or null This key passes information to the CoverSheet procedure. This string would typically be used to add a “Subject” line to the cover page.

RetryInterval
integer or null This key is a positive integer that specifies the number of minutes to wait before retrying to send a fax that failed. If the value is null, then the value is determined by the %Fax% device parameter DefaultRetryInterval (see Table 3.20); the maximum value is 60 minutes.

RevertToRaster
boolean This entry is ignored by the setpagedevice operator but used by the faxsendps operator to decide what to do when the receiving machine refuses to accept a PostScript language transmission. If true, the PostScript language job will be imaged locally and a rasterized fax transmission will be made instead. If false, the job will fail. The default value is true.

SenderID string or null This key contains a string that uniquely identifies the sender within the organization (at that fax number).

SenderMailStop
string or null This key contains information helpful for hand delivery of a return fax message. For example, Mail Stop 43A, Bldg. 2.

SenderName
string or null This key indicates the document’s sender; for example, Dr. Jane Green. A null value for SenderName leads the code to seek an alternative non-null value to store in the job log. The first alternative is to use the value of SenderOrg. If SenderOrg has a null value then CallerID is used. If this too is null, then the value of the %Fax% device parameter ID is used.

SenderOrg
string or null This key indicates the sender’s company or organization name. If SenderOrg is null, then the value of SenderName is used. If SenderName is null, the value of the %Fax% device parameter ID is used (see Table 3.20).

SenderPhone
string or null This key indicates the sender’s voice telephone number.

TrimWhite boolean If TrimWhite is true when preparing a raster transmission, then white space at the top and bottom of pages will be removed before the pages are transmitted. This can result in shorter phone calls, but also a mixture of page lengths. The default is false.
2.2.5 CoverSheet, Confirmation, and PageCaption Procedures

The CoverSheet, Confirmation, and PageCaption procedures can be used to customize fax jobs. Should a user not wish to define these, simple default procedures are supplied. It is assumed that these procedures will make no change to the global state that would effect the subsequent printed appearance of the PostScript language job.

The CoverSheet, Confirmation, and PageCaption procedures are called with no parameters. However, there will be two dictionaries on the dictionary stack which provide the information required for these procedures to do their work.

The first dictionary is the FaxOptions dictionary. Since the Confirmation procedure may be run long after the fax job was processed and since the OutputDevice will have been set to Printer typically before this procedure is run, it is actually a copy of the FaxOptions dictionary from when the fax job was processed that is present on the stack. This dictionary has the PageCaption and CoverSheet entries removed.

The second dictionary contains at least the following key-value entries:

Table 2.4 Entries in a dictionary passed to custom fax job procedures

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalleeID</td>
<td>string</td>
<td>This string is defined by the CCITT (1988) fax protocol as a string of up to 20 characters with which the callee can identify himself to the caller. It is transmitted between the two stations when they first handshake with each other. It is not present in the dictionaries provided to the CoverSheet and PageCaption procedures. For multiple call transmissions, the value determined on the first phone call is used.</td>
</tr>
<tr>
<td>CallLength</td>
<td>integer</td>
<td>This key provides the number of seconds that the transmission session lasted. It is not present in the dictionaries provided to the CoverSheet and PageCaption procedures.</td>
</tr>
<tr>
<td>CoverType</td>
<td>integer</td>
<td>This key is only meaningful to the CoverSheet procedures. It has a value indicating the type of cover needed.</td>
</tr>
<tr>
<td>CurrentPageNo</td>
<td>integer</td>
<td>This is the number of the page currently being prepared. This value can be used for generating page captions. Cover sheets are not included in this running count.</td>
</tr>
</tbody>
</table>

Value | Meaning  
---|---
0 | front cover  
1 | back cover  

2.2 PostScript Language Interface to Fax 25
ErrorArray

array of strings  This array describes particular error conditions. It is indexed by ErrorIndex.

ErrorIndex  integer  This integer can be used to retrieve a string describing a particular error condition from the ErrorArray.

IncludesFinalPage  boolean  This key indicates whether this session is the last and final transmission session for this job. If true, then this session includes the last page of the overall job, if false, it does not. For more information on transmission sessions, see InitialPage and LimitPage.

InitialPage and LimitPage  integer  This key determines which pages of an overall job have been or will be sent in this transmission. The job may get broken into more than one distinct transmission session due to errors on the telephone line, overruns or under-runs, etc. If cover pages are being generated, then each distinct transmission will have its own cover sheet. The two items here can be used by the cover page procedure to figure out how many pages are in a particular transmission and where they fall with respect to other transmissions (if any). InitialPage is the page number of the first page of the session less one. (That is, it is 0 if this transmission includes the first page of the job.) LimitPage is the total number of pages in this session and all preceding sessions of this job excluding all cover sheets.

NumberOfCalls  integer  This key specifies the number of separate telephone calls used to make the transmission. It is not meaningful in the dictionaries provided to the CoverSheet and PageCaption procedures.

PagesSent  integer  This key specifies a count of the number of pages sent to the destination fax machine. It is not present in the dictionaries provided to the CoverSheet and PageCaption procedures.

SendPostScript  boolean  This key indicates whether the job is a PostScript language file transmission. A value of true indicates that it is; false indicates that it is not.

TimeSent  array of integers  This array contains date and time information.

<table>
<thead>
<tr>
<th>Index</th>
<th>Meaning</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Year</td>
<td>[1980-2079]</td>
</tr>
<tr>
<td>1</td>
<td>Month</td>
<td>[1-12]</td>
</tr>
<tr>
<td>2</td>
<td>Day</td>
<td>[1-31]</td>
</tr>
<tr>
<td>3</td>
<td>Hour</td>
<td>[0-23]</td>
</tr>
<tr>
<td>4</td>
<td>Minute</td>
<td>[0-59]</td>
</tr>
<tr>
<td>5</td>
<td>Second</td>
<td>[0-59]</td>
</tr>
</tbody>
</table>
For confirmation reports, this is the time of the first call (of possibly many) made for the job. For cover sheets and page captions, this is the time the PostScript language job was submitted and processed.

### 2.2.6 Broadcast Transmission of Faxes

The purpose of providing a broadcast capability for fax is to optimize throughput. For a raster fax, the common body is rasterized and compressed only once. For a PostScript language file transmission, the out-bound file is assembled only once. Then, in either case, the previously prepared material is transmitted multiple times – one copy to each different recipient specified in the **Copies** array. Thus, one requirement for a broadcast job to succeed is that there must be sufficient storage (RAM or disk) to hold all outgoing files that form the fax. Broadcast transmissions are not broken into multiple calls.

As described earlier, the entries in the **Copies** array are individually merged with the original **FaxOptions** dictionary or **faxsendps** argument dictionary to produce a sequence of new dictionaries which are used to direct the mailings. Where the same key is defined in both the original dictionary and the **Copies** element, the value from the **Copies** element takes precedence (except as noted in the following discussion).

When raster is being sent, most of the **FaxOptions** entries provide mailing information and thereby determine how the fax is sent; however, a few of the entries actually effect what is to be sent. For those entries which effect the common pages to be sent, the values in the original **FaxOptions** dictionary prevail and the ones in the **Copies** dictionaries are ignored. The entries in this category are: **FaxType**, **TrimWhite**, and **PageCaption**.

The **PageCaption** procedure designated by the original **FaxOptions** will be used to place captions on the common pages (or no captions, if that is desired). Two dictionaries are placed on the dictionary stack before calling the **PageCaption** procedure. One of these is a **FaxOptions** dictionary. If broadcast pages are being prepared, the original **FaxOptions** will be used. This means that accesses to items such as **RecipientName** and **RecipientPhone** will retrieve the values from this dictionary and not from any of the merged dictionaries. Thus, giving these keys values like **(Distribution)** or **(Mailing List)**, makes sense if the **PageCaption** procedure will be using them.

If a raster is broadcast, individual cover sheets will be produced for each recipient, as determined by the **CoverSheet** procedures in the merged **FaxOptions** dictionaries. These cover sheets are imaged separately from the common pages and may contain recipient-specific information derived from the merged **FaxOptions** dictionaries.
When a PostScript language file is being broadcast, the first part of the common file sent to all recipients contains code to reproduce a dictionary. This dictionary is used by the receivers to create a cover page. The dictionary that is coded and sent is the argument dictionary passed to the fax-sendps operator, not one of the merged dictionaries.

Different Copies entries may have different values for the RevertToRaster key, just as they may each have a different PostScriptPassword. When a job asks to broadcast a PostScript language file, that is all that is initially prepared. Transmission starts and proceeds until some recipient requires raster. At this point, rasterization and compression will begin. The phoning and transmitting (of a PostScript language file) continues at the same time. If other recipients are found which demand raster, they are recorded so that when the common rasterization is performed, a new phone call can be placed. Cover sheets are imaged separately, as needed.

The value of nPages in different Copies dictionaries may vary, although this would seem to be illogical. The value of nPages is irrelevant in the context of broadcast anyway since the entire job must be rasterized before any transmissions take place. It is only used if transmission must start before imaging is done.

CoverSheetOnly is another key whose values may vary. It may have different values in different dictionaries. However, this could mean that as a result of the broadcast, some recipients will get one page (a cover sheet) and others will not even be called.

### 2.2.7 Some Sample Fax Jobs

This section presents several example PostScript fax jobs. These examples show some of the major features of the job interface.

**A Simple Example Sending a Raster File**

This snippet of PostScript language code produces a cover page (if the value of the %Fax% parameter DefaultCoversOn is true; see Table 3.20) and the document page on the fax machine answering the phone at 415-555-3710.

```
2 dict dup
begin
 /OutputDevice /Fax def
 /FaxOptions 10 dict dup begin
   /DialCallee (T9,1-415-555-3710) def
   /RecipientName (Joe Smith) def
   /FaxType 0 def
def
end
def
end
setpagedevice
```
% now comes the real document
/Bookman-Light findfont 20 scalefont setfont
100 400 moveto (Hello, world!) show
showpage

A transmission report will also be produced on the local printer.

**An Example with User-defined Procedures**

This is an extension of the previous example, and shows the use of custom cover sheet, page caption, and confirmation procedures.

2 dict dup
begin
  /OutputDevice /Fax def
  /FaxOptions 10 dict dup begin
    /DialCallee (T9,1-415-555-3710) def
    /RecipientName (Joe Smith) def

    % A simple coversheet
    /CoverSheet {
      /Times-Roman findfont
      40 scalefont setfont
      200 400 moveto
      (FAX Cover) show
    } def

    % A different page caption
    /PageCaption {
      /Times-Roman findfont
      8 scalefont setfont

      % put caption at very top of page
      300 755 moveto

      (Communicate by Fax!) show
    } def

    % Confirmation report - prints locally
    /Confirmation {
      /Times-Roman findfont
      40 scalefont setfont
      200 400 moveto
      (Fax Transmission Report) show
      200 200 moveto
      (Pages: ) show

      % access one of the dictionaries
      PagesSent 5 string cvs show

      showpage
    } def
end
def
end
setpagedevice

% now comes the real document
/Bookman-Light findfont 20 scalefont setfont
100 400 moveto (Hello, world!) show
showpage

A Simple Example Sending a PostScript Language File

This example sends the same fax message as the first example but does it by sending the PostScript language file instead of a rasterized image.

currentfile
10 dict dup begin
   /DialCallee (T9,1-415-555-3710) def
   /RecipientName (Joe Smith) def
   /FaxType 0 def
   /RevertToRaster true def
end

/FaxOps /ProcSet findresource /faxsendps get exec

% now comes the real document
/Bookman-Light findfont 20 scalefont setfont
100 400 moveto (Hello, world!) show
showpage

Note that if the fax machine being called is not willing to accept a PostScript language file (or if it requests a password, which the above code has not provided), the fax printer will hang up, image the page locally, and then transmit the rasterized page with a second phone call.

2.3 Envelope Orientation in User Space

This section describes how default user space is oriented relative to the flap on an envelope. This discussion assumes that the Install procedure does not alter the default transformation matrix.

If the PageSize value is portrait ([width height] with width < height), then default user space is set up so that the origin is on the opposite edge of the envelope from the flap and in the diagonally opposite corner from the return address (on a U.S. business envelope). The default user space is set up this way regardless of how envelopes are fed into the printer on a particular product.

Figure 2.1 illustrates two envelopes: one with its flap along the long edge of the envelope, and one with its flap along the short edge of the envelope. The dashed line indicates that the flap is on the side of the envelope facing down.
If the flap is along the long edge of the envelope, then default user space for a portrait `PageSize` is set up as in panel A of Figure 2.1.

If the flap is along the short edge of the envelope, then the default user space for a portrait `PageSize` is set up as in panel B of Figure 2.1.

**Figure 2.1** Default user space orientation for a portrait envelope `PageSize` values. For landscape `PageSize` values ( `[width height]` with `width > height`), the orientation of default user space is defined relative to the orientation for portrait `PageSize` values. This relationship is described in Table 4.10 in section 4.11 of the *PostScript Language Reference Manual, Second Edition.*

### 2.4 Errors Generated by Page Device Parameters

In addition to a `configurationerror`, the `setpagedevice` operator can generate a `typecheck`, `rangecheck`, or `invalidaccess` error under certain conditions.

If a feature is unknown for a product, then policy is invoked for that feature, without checking the type of the value. Therefore, the only error that can be generated for unknown features is a `configurationerror`, and only if the policy specifies that this is to be done. For most products, the default policy for unknown features is to ignore them.

#### 2.4.1 `typecheck` Errors

A `typecheck` error is generated if:
• The type of the value for a feature is not one of the acceptable types for that feature, or a component value within a compound value is not the correct type. Each of the following examples would generate a typecheck error:

```
<</BeginPage 4>> setpagedevice
This example generates a typecheck error.
```

```
<</Margins [0 true]>> setpagedevice
This example generates a typecheck error.
```

```
<</InputAttributes << 0 23 >> >> setpagedevice
This example generates a typecheck error.
```

• A literal array is given for a value that should be a procedure. However, an executable array is acceptable wherever an array value is expected. Packed arrays are always acceptable wherever an array is acceptable. The first two examples below would generate a typecheck error, and the third would not:

```
<</Install [2 3 4]>> setpagedevice
This example generates a typecheck error.
```

```
<</Policies <</PolicyReport [5 6 7]>> setpagedevice
This example generates a typecheck error.
```

```
<</PageSize [612 792]>> setpagedevice
This example is correct.
```

• The operand to `setpagedevice` is not a dictionary. The following example would generate a typecheck error:

```
true setpagedevice
This example generates a typecheck error.
```

2.4.2 rangecheck Errors

A rangecheck error is generated if:

• An array value of the wrong length is given, either as the value for a feature, or as a component of a value within a compound value. Each of the following examples would generate a rangecheck error:

```
<</HWResolution [300]>> setpagedevice
This example generates a rangecheck error.
```

```
<</InputAttributes << 0 << /PageSize [600 700 800] >> >> setpagedevice
This example generates a rangecheck error.
```
• A value of the right type, but beyond the acceptable range of values, is given either as the value for a feature, or as a component of a value within a compound value. Each of the following examples would generate a rangecheck error:

```
<< /PreRenderingEnhanceDetails << /Type –1 >> >> setpagedevice
```
This example generates a rangecheck error.

```
<< /Jog 10 >> setpagedevice
```
This example generates a rangecheck error if Jog is known.

### 2.4.3 invalidaccess Errors

An invalidaccess error is generated if:

• A string, array, or dictionary value is given whose access is more restrictive than read-only, either as the value for a feature or as a component value within a compound value. An exception is that for values that are procedures, the value can be execute-only. The first two examples below would generate invalidaccess errors; the third would not:

```
<< /MediaColor (blue) noaccess >> setpagedevice
```
This example generates an invalidaccess error.

```
<< /PageSize {612 792} executeonly >> setpagedevice
```
This example generates an invalidaccess error.

```
<< /BeginPage {pop} executeonly >> setpagedevice
```
This example is correct.

• The operand to setpagedevice is a dictionary whose access is more restrictive than read-only. The following example would generate an invalidaccess error:

```
<< /PageSize [612 792] >> noaccess setpagedevice
```
This example generates an invalidaccess error.
CHAPTER 3

Interpreter Parameters

The various interpreter parameters control the operation and behavior of the PostScript interpreter. Many of them have to do with allocation of memory and other resources for specific purposes. For example, there are parameters to control the maximum amount of memory used for VM, font cache, and halftone screens. Some input/output devices have parameters that control the behavior of each device individually.

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

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

Refer to the PostScript Language Reference Manual, Second Edition, Chapter 8, for descriptions of these operators, and to Appendix C in the same manual for further information about interpreter parameters.

3.1 Two Kinds of Unencapsulated Jobs

An unencapsulated job is entered by executing the Level 2 operator, start-job, or the Level 1 operator, exitserver. These operators require a password to be presented. The password must be equal to the value of either the StartJobPassword or the SystemParamsPassword system parameter. If the password is equal to the value of StartJobPassword, an ordinary unencapsulated job is started (see section 3.7.7 of the PostScript Language Reference Manual, Second Edition). If the password is equal to the value of SystemParamsPassword, a system administrator job is started. (If the SystemParamsPassword is a zero-length string or has never been set, every unencapsulated job is a system administrator job.)
3.2 Passwords for System and Device Parameters

The system parameters **StartJobPassword** and the **SystemParamsPassword** are explained in section C.3.1, “Passwords” of the *PostScript Language Reference Manual, Second Edition*. Section C.4 makes the statement “**setdevparams** is very similar to **setsystemparams**; the same restrictions apply.” This needs to be clarified a little. When setting device parameters, most but not all will require a password equal to the **SystemParamsPassword**. Also, there is one system parameter that does not require a password. The exceptions to the rules are as follows.

- The **FactoryDefaults** system parameter does not require a password if **FactoryDefaults** is the only entry in the dictionary passed to **setsystemparams**. If the only other key in the dictionary is the password, it will be ignored. This is necessary so that if the **SystemParamsPassword** has been forgotten, there will still be a way to un-set it (see **FactoryDefaults** described in Table 3.2).

- The device parameters **Interpreter** and **Protocol** found in device sets of type /Communications do not require a password if one or both are the only entries in the dictionary passed to **setdevparams**. If the only additional key in the dictionary is the password, it will be ignored (see **Interpreter** described in Table 3.5 and **Protocol** described in Table 3.6).

3.3 User Parameters

Any PostScript language program can set user parameters during job execution; no password is required. The initial value of user parameters when the printer is turned on for the first time is product dependent.

Unless otherwise specified, all user parameters are subject to **save** and **restore**. (At this time, **JobTimeout** is the only parameter that does not obey **save** and **restore**.) This means that if an unencapsulated job changes user parameters, these new values will be the initial values for subsequent encapsulated jobs. There are exceptions to this generalization. For a system parameter whose name is the same as a user parameter, the value of the system parameter is used to initialize the corresponding user parameter at the beginning of each job. In any case, changes made to any user parameter by an encapsulated job have no affect on the initial value of user parameters for subsequent jobs.

User parameters are maintained on a per context basis in environments that support multiple contexts.

The following user parameters are described in Table C.1, Appendix C of the *PostScript Language Reference Manual, Second Edition*. The description of these parameters is unchanged.
Each user parameter is identified by a key, which is always a name object. The value of the parameter is usually an integer.

Table 3.1 describes user parameters that have been defined or amended since the publication of the *PostScript Language Reference Manual, Second Edition*.

Note   This symbol is used in Table 3.1: ‡ = this key is typically present in all job server (i.e., printer) implementations.

Table 3.1 *User parameters*

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
</table>
| AccurateScreens      | boolean | This parameter controls whether the accurate screen algorithm will be used during subsequent executions of the *setscreen* and *setcolorscreen* operators. This parameter has no affect on screens established by *sethalftone*. See section 6.4.4 of the *PostScript Language Reference Manual, Second Edition*, for a description of accurate screening for the *sethalftone* operator.  
Legal values: true, false  
Errors: none |
| JobName†             | string  | This parameter establishes string as the name of the current job. If defined as a non-zero length string, status responses generated during the remainder of the current job will include a job field that reports the text of this string. The characters should be within the ASCII printable range, because this information is transmitted across arbitrary communications channels and is intended for display to users.  
Legal values: Any sequence of byte values up to an implementation-dependent maximum length. However, it should not contain the characters ‘;’ or ‘]’ because they would disrupt the syntax of status messages. If the maximum length is exceeded, the string will be truncated.  
Errors: limitcheck, typecheck |
| JobTimeout           | integer | Setting *JobTimeout* to a positive value establishes this value as the current job timeout, the number of seconds a job is allowed to execute before it is aborted and a PostScript language timeout error is generated. The current value is decremented during the job, and 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 as part of the job execution time. Setting this parameter to 0 disables job time-out altogether.

**JobTimeout** is not subject to save and restore. It is initialized to the value of the **JobTimeout** system parameter at the beginning of each job.

**Legal values:** Any non-negative integer.

**Errors:** typecheck

---

**WaitTimeout** integer

This parameter indicates the current wait timeout which is the number of seconds the interpreter waits to receive additional characters from the host before it aborts the current job by executing a PostScript language **timeout** error. A value of 0 indicates an infinite timeout. This parameter is initialized to the value of the **WaitTimeout** system parameter at the beginning of each job.

**Legal values:** Any non-negative integer.

**Errors:** typecheck

---

### 3.4 System Parameters

In general, setting system parameters requires a password. System parameter values persist across jobs. (Depending upon the product, some system parameters are stored in non-volatile memory and are persistent across restarts of the interpreter.)

System parameters are global to the PostScript environment and, in particular, are not maintained on a per context basis in the environments that support multiple contexts. The initial value of system parameters when the device is turned on for the first time and which parameters are stored in non-volatile memory are product dependent.

Some system parameters are **read-only**: that is, they are returned by **currentsystemparams**, but any attempt to change one using **setsystemparams** has no affect. Other parameters are **write-only**. They can be set by **setsystemparams**, but are not returned by **currentsystemparams**.

Each system parameter is identified by a key, which is always a name object. The following system parameters are described in the *PostScript Language Reference Manual, Second Edition*. The description of these parameters is unchanged.
Note  In the following lists, as well as in Table 3.2, this symbol is used:
‡ = this key is typically present in all job server (i.e., printer) implementations.

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>BuildTime</td>
<td>integer</td>
<td>(Read-only) A time stamp identifying a specific build of the PostScript interpreter. The values returned by BuildTime on two different products need not be comparable, and in general, BuildTime should only be interpreted in conjunction with the manufacturer’s product documentation. Legal values: Any integer. Errors: none</td>
</tr>
<tr>
<td>CurInputDevice</td>
<td>string</td>
<td>(Read-only) This parameter indicates the name of the communications device corresponding to the current input file for the currently executing PostScript language program. The string that is returned corresponds to the communications device parameter set name whose values are normally stored in RAM: for example, (%Serial%). For more information on communication devices, see section 3.5.2, “Communication Devices.” Legal values: A string containing a communications device name. Errors: none</td>
</tr>
<tr>
<td>CurOutputDevice</td>
<td>string</td>
<td>(Read-only) This parameter indicates the name of the communications device corresponding to the current output file for the currently executing PostScript language program. The string that is returned corresponds to the...</td>
</tr>
</tbody>
</table>
communications device parameter set whose values are normally stored in RAM: for example, (%Serial%). For more information on %Serial% devices, see section 3.5.2, “Communication Devices.”

**Legal values:** A string containing a communications device name.

**Errors:** none

**CurSourceList**

integer  (Read-only) This parameter indicates the number of bytes currently occupied by source lists. A source list holds the internal data representation for sampled image source data and uncached character pixel arrays.

**Legal values:** Any non-negative integer.

**Errors:** none

**CurStoredFontCache**

integer  (Read-only) This parameter indicates the number of bytes that the storage device font cache currently occupies.

**Legal values:** Any positive integer.

**Errors:** none

**CurStoredScreenCache**

integer  (Read-only) This parameter indicates the number of bytes currently used for screen files on the storage device which includes the currently active screens.

**Legal values:** Any positive integer.

**Errors:** none

**DoStartPage**

boolean This parameter indicates whether the start page should print during system initialization. The start page prints if the value of **DoStartPage** is **true** during system initialization.

**Legal values:** **true, false**

**Errors:** typecheck

**FactoryDefaults**

boolean This parameter is usually **false**. Setting it to **true** and immediately turning off the printer causes all non-volatile parameters to revert to factory default values at the next power-on. The job that sets **FactoryDefaults** to **true** must be the last job executed before power-off; otherwise, the request is ignored. This required physical interaction reduces the chance of malicious jobs resetting the device to factory defaults.
A password is not required in the dictionary passed to `setSystemParams` if `FactoryDefaults` is the only entry in the dictionary. This allows the factory defaults to be reestablished even though the system parameters password might have become corrupted. Note that the passwords are among those reset by this operation.

The exact collection of parameters reset to factory defaults by this action is product dependent. In most products, `PageCount` is not reset.

**Legal values:** `true`, `false`

**Errors:** `typecheck`

### FatalErrorAddress

A fatal system software error causes a PostScript device to stop execution and, in most products, to restart the PostScript interpreter. Before execution is stopped, the address at which the error occurs is stored in the parameter `FatalErrorAddress` and also is transmitted to the host over the communications channel. A non-zero value of this parameter indicates that a fatal system error has occurred earlier. On some products, if this value is non-zero during system initialization, the address is printed on the start page or possibly on a separate page.

**Legal values:** Any integer.

**Errors:** `none`

### FontResourceDir

This parameter controls the location of external fonts. Fonts are resources in PostScript Level 2. The `Font` category implementation concatenates the `FontResourceDir` and the font name to get the external location of the font. For example, if the `FontResourceDir` were `(Resource/Font/)`, then the Times-Roman resource of the `Font` category would be in `(Resource/Font/Times-Roman)`.

This parameter is provided separately from the `GenericResourceDir` system parameter to allow backward compatibility with applications that expect fonts to be located under `(fonts/)`. In such a case, `FontResourceDir` should be set to `(fonts/)`

Note that applications and users should access external fonts only through the resource operators or `findfont` or, if necessary to access them as files, through `ResourceFileName`. (See the PostScript Language Reference)
The above parameter should be used only to control the location of external fonts by the resource management mechanism.

**Legal values:** Any string with non-null characters.

**Errors:** limitcheck, typecheck

**GenericResourceDir** ‡
**GenericResourcePathSep** ‡

strings These parameters control the location of external resources for the **Generic** category and all categories based upon it (currently **Category**, **Encoding**, **Form**, **Pattern**, **ProcSet**, **ColorSpace**, **Halftone**, and **ColorRendering**). The **Generic** category implementation concatenates the **GenericResourceDir**, the category name, the **GenericResourcePathSep**, and the resource name to get the external location of the resource. For example, if the **GenericResourceDir** and **GenericResourcePathSep** were (Resource/) and (/), respectively, then the AdobeLogo resource of the **Pattern** category would be in Resource/Pattern/AdobeLogo.

The **GenericResourceDir** should be an absolute path, that is, a path beginning at the root of the storage device. It must contain any trailing path separator. It should include a storage device (e.g., %os%) if only a single device is to be considered, or should omit the device if all searchable devices are to be considered. If there is a device specifically for generically managed resources (e.g., %GenericResource%) that may access resources through a network server or along a search path, then **GenericResourceDir** should be set to that device. Resource files are expected to be in subdirectories with names the same as category names. The resource file name should be the same as the name of the resource it defines. In the above example, the file named Resource/Pattern/AdobeLogo should contain a PostScript language program which, when run, will define the AdobeLogo instance in the **Pattern** resource category.

Note that applications and users should access external resources only through the resource operators or, if necessary to access them as files, through **ResourceFileName**. (See the **PostScript Language Reference Manual, Second Edition**, section 3.9, “Named Resources.”) The above parameters should be used only to control the location of external resources by the resource management mechanism.
For products with no external resources (and, presumably, no file systems), \texttt{GenericResourceDir} should be set to \texttt{(%null)}. This mechanism can also be used by site administrators to temporarily disable access to external resources.

\textbf{Legal values:} Any string with non-null characters.

\textbf{Errors:} \texttt{limitcheck, typecheck}

\textbf{JobTimeout} \hspace{1cm} \texttt{integer} \hspace{1cm} This parameter indicates the value in seconds to which the user parameter \texttt{JobTimeout} will be initialized at the beginning of each job. Trying to set the system parameter \texttt{JobTimeout} to a negative value will be ignored and the previous setting of \texttt{JobTimeout} will be used. A value of 0 indicates that the timeout will be infinite. Trying to set a number between 1 and 14 will result in 15 being set (i.e., 15 is the minimum value). The reason for the minimum value of 15 is that if small values were allowed, this might prevent a subsequent job from setting \texttt{JobTimeout} to another value successfully.

\textbf{Legal values:} 0 or any integer greater than or equal to 15.

\textbf{Errors:} \texttt{typecheck}

\textbf{LicenseID} \hspace{1cm} \texttt{string} \hspace{1cm} This parameter contains the Adobe-assigned license identifier. Its value is unique to each product.

\textbf{Legal values:} Any string of non-null characters.

\textbf{Errors:} \texttt{limitcheck, typecheck}

\textbf{MaxHWRenderingBuffer} \hspace{1cm} \texttt{integer} \hspace{1cm} This parameter indicates the amount of memory, in bytes, to reserve for use by a hardware rendering device, such as PixelBurst\textsuperscript{TM}, to store display list data. The memory is permanently allocated during system initialization. If the value being set is outside of the legal range, \texttt{MaxHWRenderingBuffer} is set to the nearest acceptable value. The minimum value meets the requirements of the rendering device and the maximum value is an amount that will not jeopardize the execution of a PostScript language job.

\textbf{Legal Values:} Any positive integer (product dependent, typically 8192 or greater).

\textbf{Errors:} none
MaxImageBuffer‡
integer  This parameter indicates the maximum number of bytes that can be utilized for a single image buffer. An image buffer holds an internal data representation for sampled image source data. The parameter may be rounded by the interpreter if a requested value is out of range.

Legal Values:  Any integer.

Errors:  typecheck

MaxRasterMemory
integer  This parameter indicates the largest amount of memory, in bytes, that may be allocated to the frame buffer. This parameter may be used to limit the amount of raster memory; unused raster memory is available for use as VM. Thus, MaxRasterMemory allows the user to trade-off raster memory allocation (which will allow larger page sizes and higher resolutions) against VM (which will allow more downloaded fonts and the production of more complex pages). MaxRasterMemory is consulted only during system initialization; any changes to the value of the parameter will not take effect until then.

Legal values:  Product dependent.

Errors:  typecheck

MaxSourceList‡
integer  This parameter indicates the maximum number of bytes that can be utilized for source lists. A source list holds internal data representation for sampled image source data and uncached character pixel arrays. This parameter may be rounded by the interpreter if a requested value is out of range.

Legal values:  Any integer.

Errors:  typecheck

MaxStoredFontCache
integer  This parameter defines the maximum number of bytes that the storage device font cache can occupy on the chosen storage device (such as the disk). Setting MaxStoredFontCache to 0 has the affect of turning off stored caching. Setting MaxStoredFontCache to -1 (or to a value too large), sets the number of bytes that the font cache can occupy to the logical size of the storage device. If the logical size of the storage device is not known, an implementation dependent value will be used.

Legal Values:  -1, 0, or any positive integer

Errors:  typecheck
MaxStoredScreenCache
integer  This parameter defines the maximum number of bytes that the storage device screen cache can occupy on the chosen storage device. Setting MaxStoredScreenCache to 0 turns off stored caching. Setting MaxStoredScreenCache to a negative value (or to a positive value too large), sets the number of bytes that the screen cache can occupy to the logical size of the storage device. If the logical size of the storage device is not known, an implementation dependent value will be used.

Legal Values:  0 or any positive or negative integer

Errors:  typecheck

PageCount integer  (Read-only) This parameter indicates the number of pages that have successfully printed since manufacture. In most products, PageCount will not reset at a user request to return to factory defaults. However, PageCount will be reset if the non-volatile memory in which it is stored has been corrupted.

Legal values:  Any non-negative integer.

Errors:  none

PrinterName string  This parameter establishes string as the current name of the device. If the device is on a network, this name might be used by the system as part of a name identifier for the device considered as a node on the network. PrinterName is usually printed on the start page and so it should consist of printable characters, although this is not required. Setting this parameter to a zero length string causes PrinterName to be set to the value of the product string in systemdict.

Legal values:  Any string of 32 or fewer non-null characters.

Errors:  limitcheck, typecheck

RamSize integer  (Read-only) This parameter indicates in bytes the amount of installed RAM available to the product. In some cases, this value might be less than the total amount of installed RAM. For example, the system diagnostics might have determined that certain banks of RAM are defective and would consider them unavailable.

Legal values:  Any non-negative integer.

Errors:  none
**Revision**  
integer  
*(Read-only)* This parameter designates the current revision level of the product in which the PostScript interpreter is running. Each product has its own numbering system for revisions, independent of those of any other product. The value is identical to the value of the integer `revision` in `systemdict`.

**Legal values:** Any integer.

**Errors:** none

**StartJobPassword**  
‡string  
If a program starts an unencapsulated job using `startjob` or `exitserver`, and the password it presents to that operator is the value of `StartJobPassword`, then the subsequent unencapsulated job will need to present a password equal to the `SystemParamsPassword` each time `setsystemparams`, `setdevparams`, or other system administrator operations are invoked.

**Legal values:** Any string of 32 or fewer non-null characters.

**Errors:** limitcheck, typecheck

**StartupMode**  
integer  
This parameter controls whether the system start file (Sys/Start) or some other start-up procedure should be executed during system initialization. The Sys/Start file executes if the value of `StartupMode` is 1 during system initialization. If the `StartupMode` value is 0, no special start-up procedures are run during system initialization. Other values of `StartupMode` can occur in specific products and result in product-dependent start-up procedure execution.

**Legal values:** Product dependent, but restricted to values between 0 and 255.

**Errors:** typecheck

**SystemParamsPassword**  
‡string  
If a program starts an unencapsulated job using `startjob` or `exitserver`, and the password it presents to that operator is the value of `SystemParamsPassword`, then the subsequent unencapsulated job is permitted to invoke `setsystemparams`, `setdevparams`, or other system administrator operations without presenting a password each time. This extends to Level 1 compatibility operators that change system parameters but provide no means to present a password.

**Legal values:** Any string of 32 or fewer non-null characters.

**Errors:** limitcheck, typecheck

**ValidNV**  
boolean  
*(Read-only)* This parameter indicates whether non-volatile memory is currently used to store persistent parameters. During system initialization, if non-volatile memory is corrupt, factory defaults are reestablished. If further
testing indicates that non-volatile memory is defective, it will not be used, and ValidNV will be false. Otherwise, ValidNV will be true. In many products, if non-volatile memory is defective, it is emulated in RAM. The operating behavior is the same, except that persistent parameter values are lost when the printer is powered off or restarted and factory defaults are used at power-on.

Legal values: true, false

Errors: none

**WaitTimeout**‡ integer

This parameter indicates the value in seconds to which the user parameter WaitTimeout will be initialized at the beginning of each job. A value of 0 indicates that the timeout is infinite. Trying to set the system parameter WaitTimeout to a negative value will be ignored and the previous setting of WaitTimeout will be used.

Legal values: 0 or any positive integer.

Errors: typecheck

### 3.5 Device Parameters

Device parameters are set using the operator `setdevparams` and are read using the operator `currentdevparams`. Device parameters are similar to system parameters in that they require a password (if the `SystemParams-Password` is set), are global to the PostScript environment, and persist across jobs. As with system parameters, some of these parameters may be stored persistently in non-volatile memory.

Device parameters are subdivided into sets that correspond to a particular device (%Serial%, %disk2%, etc.). More generally, “device” in this context really means “named parameter set.” Each named device known to the `currentdevparams/setdevparams` operators corresponds to an instance of the IDevice resource category and can represents a set of parameters describing the configuration of a physical or logical communication channel, storage device, hardware device, or software entity such as a language emulator. (See section C.4 of the Postscript Language Reference Manual, Second Edition for more details.) Even if two products have the same named device, the parameters in the set might differ, for example, because the hardware support for that device differs.
3.5.1 Device Parameter Dependencies

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

For example, in the serial communication device set there is an Interpreter and a Protocol parameter. The Interpreter parameter determines which page description language will be used for an incoming job on that channel. The Protocol parameter determines the communication protocol used to send and receive data. Protocol can be set to /Binary, /Normal, /Raw, and /TBCP. The serial channel cannot be configured to have Protocol set to /Raw and Interpreter set to /PostScript. This would be an illegal combination of device parameters. This condition is termed a configuration error. A PostScript language error (configurationerror) occurs if setdevparams attempts to establish such an illegal configuration.

Most configuration dependencies are between parameters in the same device parameter set. There is a dependency among all communications devices, however, that requires at least one of the communications channels to be On and Enabled. There might also be cases where certain device parameter sets have interdependencies. For example, if both LocalTalk and a serial channel share the same hardware port on a printer, there is a requirement that at most one of these channels be On. If one channel is already On and the other is turned On, the first is turned off and disabled.

3.5.2 Communication Devices

Printers provide a variety of ways to set up communications parameters, including front panels, hardware switches, and PostScript language operators and procedures. The scheme described in this section provides a generic model for setting communications parameters. This model works across a variety of products and enables PostScript language spoolers and utilities to use the same model when reading and writing communication device parameters.

Communications Parameter Sets

A raster output device typically has several hardware ports for communication. Each hardware port has a named communications parameter device set associated with it. For example, in a certain large family of controllers, the parallel port is associated with the parameter device set named %Parallel%. Channel A of the serial communications controller (SCC) chip, which usually is wired to a 25-pin connector, is associated with the parameter device set named %Serial%. Channel B of the SCC chip, which usually is wired to either an 8-pin or a 9-pin connector, is associated with the parameter device set named %SerialB%. In some cases, two device sets may be associated
with the same port. For example, the %SerialB% device and the %Local-
Talk% device can both be associated with the channel B port of the SCC
chip.

For any given communications device set, there are three sets of parameters. If the name of the device is %CommName%, the names of the three parameter sets are %CommName_NV%, %CommName%, and %CommName_Pending%. For example, in a printer with an SCC chip and a parallel port, the following parameter sets probably would be available:

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

Considering only one communications device set, its three parameter sets have the following general characteristics:

- %CommName_NV% values usually are stored in non-volatile memory.
- %CommName% values usually are stored in RAM and do not persist when the printer is powered off.
- %CommName_Pending% is a read-only parameter set whose values will be used to configure the communications hardware at the beginning of the next job. This parameter set reflects either the current values of some writable 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 below.

- When there are multiple instances of a certain channel type, the naming convention is %CommName%, %CommNameB%, %CommNameC%, and so on.

The name %CommName_NV% is only a hint of actual behavior. In products with limited non-volatile memory, only some of the %CommName_NV% set parameters may actually be saved to non-volatile memory, while products with sufficient non-volatile memory typically save all writeable %CommName_NV% parameters. PostScript language utility programs need not take these differences into account. If their specific intent is to affect persistent values, they should use %CommName_NV%. The implementation will do the best it can given the amount of non-volatile memory available in the product.

There is a hierarchical relationship between these parameter sets as described below. On some products these three sets may not be distinct from each other. The reason for the presence of the three sets on all products is to provide for a consistent model that is product independent.
Basic Hierarchy of Parameter Sets

This description begins with a simple subset of the model and progresses to more complex situations.

Figure 3.1 Relationship between the communications parameter sets

Figure 3.1 shows the basic relationship of the three parameter sets. In this figure, values written to $\%\text{CommName}\%$ are written through to $\%\text{CommName}_\text{Pending}\%$; and values written to $\%\text{CommName}_\text{NV}\%$ are written through to $\%\text{CommName}\%$ and thence to $\%\text{CommName}_\text{Pending}\%$. Beyond this, several variables exist:

- The product may have a front panel. The values set by the user at the front panel are written to $\%\text{CommName}\%$ or to $\%\text{CommName}_\text{NV}\%$ (if the values are to persist across restarting and power cycling). Some products store to only one of these sets.

- The product may have switches through which it can be directed to use either $\%\text{CommName}_\text{NV}\%$ parameter sets or built-in (hard-wired) values. (This situation is discussed at length later in this section.)

Most products do not have both a front panel and switches.

- PostScript language programs (usually spoolers or utilities) may write parameter values to $\%\text{CommName}\%$ or $\%\text{CommName}_\text{NV}\%$ (usually the former) at any time. This is true whether the output device has a front panel or has switches.

In Figure 3.1, the $\%\text{CommName}\%$ parameter set, which is in RAM and does not persist when the printer is powered off, is used in many cases (but not all) to update the $\%\text{CommName}_\text{Pending}\%$ set. Thus, on many products (those with a front panel but no switches), the $\%\text{CommName}\%$ and $\%\text{CommName}_\text{Pending}\%$ sets always have the same values and appear redundant.
The \%CommName_NV\% set usually stores the parameters in non-volatile storage. In the simple case in Figure 3.1, writing to \%CommName_NV\% writes through to \%CommName\%, which in turn writes through to \%CommName_Pending\%.

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 printer is turned off.

A front panel usually writes to \%CommName_NV\% to change the power-on parameters, although the front panel also can write to \%CommName\%.

**Multiple Non-Volatile Sets**

Complicating this picture, it is possible to have more than one non-volatile parameter set. Such sets are correctly named as follows: \%CommName_NV\%, \%CommName_NV2\%, \%CommName_NV3\%, and so on. As is the case with a single non-volatile set, these parameter sets obtain their values by being written to by a PostScript language spooler or utility.

**Figure 3.2 Communications parameters sets using NV values**

![Diagram showing switch settings and non-volatile parameter sets](image)

Figure 3.2 shows a situation in which there are three non-volatile sets. Only one of these sets can be active at any given time. The switch setting indicates which one is active. In this figure, the active set is \%CommName_NV2\%, which is indicated by the switch setting. When the switch is set to this position, or when the product is restarted or powered up 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 language job can write to any of the non-volatile parameter sets, but only if it wrote to \%CommName_NV2\% would the values 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 ones in %CommName% and %CommName_Pending% instead.

**Predetermined Parameter Values**

In addition to the switch settings that indicate which non-volatile parameter set should be used, there can also be other switch settings that short-cut this hierarchy of parameter sets and cause a predetermined set of communications parameters to be written directly to %CommName_Pending%. This situation is shown in Figure 3.3.

**Figure 3.3** Communications parameter sets using “hard wired” values

In the figure, switch positions 1 and 2 designate two such “hard-wired” parameter sets. When the switch is set to position 1, for example, PostScript language programs may still write to one of the %CommName_NV% sets or to %CommName%, but there is no effect on %CommName_Pending% unless the switch is reset to one of positions 3 through 5.

This example explains the existence of the %CommName_Pending% set as separate from the %CommName% set: it allows absolute determination of the communications parameters that will be used, no matter what other activity occurs.

Note that reading the %CommName_NV% set or the %CommName% set gives you 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 the job 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 non-volatile memory and without concern for whether a utility job, front panel, or switch is used to establish communication parameters.

As in the case 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 restarting and power cycling.


### 3.5.3 Device Parameter Tables

Tables 3.3 through 3.21 list all device parameters that are currently defined.

#### Table 3.3 Parameters common to all device parameter sets

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) This parameter designates the general category of device represented by the parameter set. Every device parameter set will contain a Type entry.</td>
</tr>
</tbody>
</table>

**Legal values:** /Communications, /Emulator, /FileSystem, /Parameters

**Errors:** none

#### Table 3.4 Parameters common to device sets of type /FileSystem, or /Communications

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasNames</td>
<td>boolean</td>
<td>(Read-only) This parameter indicates whether the device represented by the parameter set supports named files. If the device is not mounted, this parameter will have a value of false. This is defined only in device parameter sets of the Type /FileSystem or /Communications. If Type is /Communications, HasNames is always false.</td>
</tr>
</tbody>
</table>

**Legal values:** true, false

**Errors:** none
Table 3.5 Parameters typically present in parameter sets of type /Communications

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>boolean</td>
<td>This parameter designates whether data arriving on the device represented by the parameter set should be considered as a job to be scheduled for execution by the PostScript interpreter or an emulator. If Enabled is true, arriving data will be scheduled as an executable job. If Enabled is false, the data will not be scheduled as an executable job, but the device can be used directly by a job for reading and writing data. A configurationerror is generated if setting Enabled would produce either of the following situations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the same parameter set, On is false and Enabled is true.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• All communication devices have Enabled set to false.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: configurationerror, typecheck</td>
</tr>
<tr>
<td>Interpreter</td>
<td>name</td>
<td>This parameter designates the type of executable job the arriving data represents. This parameter is used only if Enabled is true. For certain communication channels there is a relationship between the Interpreter and the Protocol parameters that can result in a configurationerror. See Protocol for details.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Either Interpreter or Protocol or both can be set without a password if no other parameters are specified in the execution of setdevparams.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Interpreter value /AutoSelect is described below. For information on the other legal values, see section 3.5.8, “Emulator Parameters.”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/AutoSelect: The AutoSelect facility provides automatic and seamless switching between the available interpreters and emulators based on the input data stream. The Interpreter parameter should be set to /AutoSelect on channels that connect to hosts which alternately send PostScript language jobs, raw PCL (LaserJetIIP or LaserJetIII), and “printscreen” jobs (in the IBM PC compatible environment). Primarily it can be used on both serial and parallel communication channels or on certain other communication channels were it makes sense to do so. When using AutoSelect for a given communications channel, it is important that the underlying communication protocol is one that preserves all incoming data. In particular, for a serial or parallel channel, this implies that Protocol is set to Raw, Binary, or TBCP.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For serial and parallel communication channels, the following is true.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• AutoSelect detects interpreter boundaries and job boundaries if the value of Protocol is set to TBCP or Binary.</td>
</tr>
</tbody>
</table>
• AutoSelect detects interpreter boundaries, job boundaries, and protocol boundaries and automatically selects the protocol if the value of Protocol is set to Raw. This is the recommended setting for Protocol when using AutoSelect.

• When Interpreter is set to AutoSelect, the value of Protocol must be either Binary, Raw, or TBCP; otherwise, a configurationerror is generated.

Legal values: /PostScript, /AutoSelect, /Diablo630, /EpsonFX850, /HP7475A, /LaserJetIII, /LaserJetIP, /ProprinterXL

Errors: configurationerror, rangecheck, typecheck

On boolean This parameter designates whether the device driver for the communications device is turned on and able to receive and send data. If the parameter is true, data transmitted to the device by a host is buffered and flow control protocols are applied. Data sent to the device when this parameter is false is lost. A configurationerror is generated if setting the On parameter would produce a situation in which On is false and Enabled is true in the same parameter set.

If two communications devices share the same physical port, and setting the On parameter produces a situation in which both devices had On set to true, 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 device is not considered as a source of jobs to be scheduled, but the device can be used by a PostScript language job to send and receive data by means of the file operators.

Legal values: true, false

Errors: configurationerror, typecheck

Table 3.6 Parameters typically present in serial communications parameter sets

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud</td>
<td>integer</td>
<td>This parameter designates the baud rate on the underlying serial hardware. Normally this parameter can be set to any non-negative number; it will not be rounded. The underlying serial hardware will, however, round the baud rate to the nearest achievable value. 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 legal values.</td>
</tr>
</tbody>
</table>

Legal values: Product dependent.

Errors: rangecheck, typecheck
**CheckParity** boolean  
This parameter designates whether parity checking is done by the device on incoming data. This parameter is ignored if the value of **Parity** is **None**. If **CheckParity** is **true** and a parity error occurs, a PostScript language **ioerror** results. If **CheckParity** is **false**, no parity checking occurs.

**Legal values:**  *true, false*

**Errors:**  *typecheck*

**DataBits** integer  
This parameter designates the number of data bits per byte communicated over the channel. If this parameter is 7, the high bit of a received byte of data is set to 0. The total number of bits for each byte transmitted or received is the sum of the number of start bits (always 1), data bits, parity bits, and stop bits.

**Legal values:**  *7, 8*

**Errors:**  *rangecheck, typecheck*

**Enabled** boolean  
For the general definition of **Enabled**, see Table 3.5.

**Legal Value:**  *true, false*

**Errors:**  *configurationerror, typecheck*

**FlowControl** name  
This parameter designates the serial flow control method used between the host and the device.

*Note* Not all serial channels support all flow control modes.

Following are descriptions of the legal values of **FlowControl**:

-/Dtr: DTR and DSR hardware signals are used by the printing device and the host, respectively, to indicate to the 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: This parameter is the same as /Dtr except the active sense of the signals is reversed. A low signal indicates that data may be transmitted and a high signal indicates that data should not be transmitted.

-/EtXAck: Two characters, ETX and ACK, are reserved for flow control usage. The protocol is symmetric for printing device and host. Each sender knows an agreed upon maximum number of characters that the other side can receive. A sender may send up to this number of characters followed by an ETX. The sender may send more data only when it has received an ACK from the receiver on the other side.
/RobustXonXoff: This protocol operates similarly to the /XonXoff protocol except that periodically the interpreter will send the host an Xon if it is able to receive data.

/XonXoff: Two characters, Xon and Xoff, are reserved for flow control usage. For all Protocol settings except /Raw, the protocol is symmetric for printing device and host. If one side wishes the other to stop sending data, it sends an Xoff. When it is ready to receive data again it sends an Xon. If Protocol is set to /Raw, Xon and Xoff sent from the host to the printer are treated as data and not reserved as flow control characters. Xon and Xoff sent from the printer to the host are always treated as flow control characters.

/XonXoff2: This protocol operates similarly to the /XonXoff protocol except that the interpreter will ensure that the data channel remains in the Xon state even when the channel is Off.

Legal values: /Dtr, /DtrLow, /EtxAck, /RobustXonXoff, /XonXoff, /XonXoff2

Errors: rangecheck, typecheck

HasNames boolean (Read-only) This constant always returns a value of false. For the general definition of HasNames, see Table 3.4.

Legal Value: false

Errors: none

Interpreter name For the general definition of Interpreter, see Table 3.5.

Legal values: /PostScript, /AutoSelect, /Diablo630, /EpsonFX850, /HP7475A, /LaserJetIII, /LaserJetIP, /ProprinterXL

Errors: configurationerror, rangecheck, typecheck

On boolean For the general definition of On, see Table 3.5.

Legal Value: true, false

Errors: configurationerror, typecheck

Parity name This parameter designates the parity to be used between the host and the device. If Parity is Space or Mark, the parity bit should always be 0 or 1, respectively. If Parity is None, neither the host nor the device should send a parity bit. If Parity is Even, even parity is used. If Parity is Odd, odd parity is used. The total number of bits for each byte transmitted or received is the sum of the number of start bits (always 1), data bits, parity bits, and stop bits. Most serial devices do not support 8-bit data with either space or mark
parity, although setting the parameters in this manner does not generate a **configurationerror**. The results of this configuration, however, are unpredictable.

**Legal values:**  
/Even, /Mark, /None, /Odd, /Space

**Errors:**  
rangecheck, typecheck

**Protocol**  
This parameter indicates the communications protocol that will be used. These communications protocols are available:

- **/Binary:** In /Binary mode, an encoding scheme allows the full range of 8-bit values to be transmitted as data while also providing for certain communication functions, such as end-of-file, software flow control, abort job, status query, etc. This protocol is suitable for use with any language (for example, the PostScript language or a printer emulation). However, it is obsolete and has been superceded by /TBCP.

- **/Normal:** In /Normal mode, certain control characters are reserved as communication functions, such as end-of-file, software flow control, abort job, status query, etc. These codes cannot be carried as data. This protocol is suitable for use only when sending ASCII-encoded PostScript language jobs; it is unsuitable for PostScript language jobs containing binary data or any printer emulation jobs.

- **/Raw:** In /Raw mode, all characters are treated as data; there are no reserved characters, and none of the communication functions are available. Normally, this protocol is suitable for use only with printer emulation, not with the PostScript interpreter. However, in products that support an **Interpreter** value of /AutoSelect, protocol processing is handled by the AutoSelect facility; therefore, **Protocol** should be /Raw in that case.

- **/TBCP:** In /TBCP (Tagged Binary Communication Protocol) mode, an encoding scheme allows the full range of 8-bit values to be transmitted as data, while also providing for certain communication functions, such as end-of-file, software flow control, abort job, status query, etc. It also provides explicit begin-protocol and end-protocol sequences that permit the receiver to switch automatically between /Normal and /TBCP mode processing. This protocol is suitable for use with any language (for example, the PostScript interpreter or a printer emulation).

For more information on these protocols, see the *Adobe Communications Protocols Specification*.

A **configurationerror** is generated if setting the **Protocol** or **Interpreter** parameter would produce either of the following situations when the **Enabled** parameter is true:
• **Protocol** with a value of /Raw and **Interpreter** with a value of /PostScript.

• **Protocol** with a value of /Normal and **Interpreter** with a value other than /PostScript.

• **Protocol** with a value of /Normal and **Interpreter** with a value of /AutoSelect.

That is, PostScript language 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 doing automatic selection of interpreters and emulators, the /Normal protocol cannot be used.

Either **Protocol** or **Interpreter** or both can be set without a password if no other parameters are specified in the execution of `setdevparams`.

**Legal values:** /Binary, /Normal, /Raw, /TBCP

**Errors:** configurationerror, rangecheck, typecheck

### StopBits integer

This parameter designates the number of stop bits that will be transmitted by the serial hardware. The hardware will always be able to receive data transmitted with one or two stop bits. The total number of bits for each byte transmitted or received is the sum of the number of start bits (always 1), data bits, parity bits, and stop bits.

**Legal values:** 1, 2

**Errors:** rangecheck, typecheck

### Type name

*(Read-only)* This constant always returns a value of Communications. For the general definition of **Type**, see Table 3.3.

**Legal value:** /Communications

**Errors:** none

---

<table>
<thead>
<tr>
<th><strong>Key</strong></th>
<th><strong>Type</strong></th>
<th><strong>Semantics</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>boolean</td>
<td>For the general definition of <strong>Enabled</strong>, see Table 3.5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal Value:</strong> true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> configurationerror, typecheck</td>
</tr>
</tbody>
</table>

---

**Table 3.7** Parameters typically present in LocalTalk communications parameter sets

**3.5 Device Parameters** 59
**Filtering**

name  This parameter indicates whether the input stream needs further filtering before the data can be correctly interpreted as a page description language.

/InterpreterBased: In this mode, the input stream is filtered as necessary to conform to the language. For example, the data stream may have been sent to the printer encoded as a TBCP PostScript language job and must be decoded to a normal PostScript language job before it is passed to the interpreter (see Protocol in Table 3.6 for a description of TBCP).

/None: Pass the data unchanged to the interpreter.

Warning  In a complete AppleTalk/Macintosh environment, **Filtering** should be set to /None or you will encounter communication problems.

Legal values:  /InterpreterBased, /None

Errors:  configurationerror, rangecheck, typecheck

**HasNames**  boolean  (Read-only) This constant always returns a value of false. For the general definition of HasNames, see Table 3.4.

Legal Value:  false

Errors:  none

**Interpreter**  name  For the general definition of Interpreter, see Table 3.5.

Legal values:  /PostScript, /AutoSelect, /Diablo630, /EpsonFX850, /HP7475A, /LaserJetIII, /LaserJetIIIP, /ProprinterXL

Errors:  configurationerror, rangecheck, typecheck

**LocalTalkType**  string  This parameter represents the type piece of the AppleTalk entity name. The entity consists of three pieces: zone, type, and object, each of which is a string of 32 or fewer non-null characters. The object piece is set to the value of the PrinterName system parameter and the zone is wildcarded.

If the printer also supports EtherTalk communications, setting the LocalTalkType string will set the EtherTalkType parameter (within the %EtherTalk% parameter set) to the same value. The appletalktype compatibility operator will reflect a change to the LocalTalkType parameter. Therefore, getting the LocalTalkType parameter will always yield the same value as getting the EtherTalkType parameter and will match what is returned by the appletalktype compatibility operator.

Legal values:  Any string of 32 or fewer non-null characters.

Errors:  limitcheck, typecheck
**NodeID**  integer  *(Read-only)* This parameter represents the local network address of the device. Legal addresses are values between 1 to 254, inclusive. If the value of **NodeID** is 0, this indicates that the address has not been established. The value is used as an address hint when first establishing addresses as part of the LocalTalk protocol. As such, the parameter might not represent the actual address until that portion of the protocol is complete during initialization of the LocalTalk device.

**Legal values:**  Any integer between 0 and 254 inclusive.

**Errors:**  none

**On**  boolean  For the general definition of **On**, see Table 3.5.

**Legal Value:**  true, false

**Errors:**  configurationerror, typecheck

**Type**  name  *(Read-only)* This constant always returns a value of Communications. For the general definition of **Type**, see Table 3.3 on page 53.

**Legal value:**  /Communications

**Errors:**  none

---

**Table 3.8 Parameters typically present in EtherTalk communications parameter sets**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enabled</strong></td>
<td>boolean</td>
<td>For the general definition of <strong>Enabled</strong>, see Table 3.5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal Value:</strong> true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> configurationerror, typecheck</td>
</tr>
</tbody>
</table>

**EthernetAddress**  string  *(Read-only)* This parameter is a unique string that represents the Ethernet address of the printer. The string is of the form XX:XX:XX:XX:XX:XX where each XX represents a byte in hexadecimal.

**Legal Values:**  Any correctly formatted string of 17 characters.

**Errors:**  none

**EtherTalkType**  string  This parameter represents the type piece of the EtherTalk entity name. The entity name consists of three pieces: zone, type, and object, each of which is a string of 32 or fewer non-null characters.
If the printer also supports LocalTalk communications, setting the EtherTalkType string will set the LocalTalkType parameter (within the %LocalTalk% parameter set) to the same value. The appletalktype compatibility operator will reflect a change to the EtherTalkType parameter. Therefore, getting the EtherTalkType parameter will always yield the same value as getting the LocalTalkType parameter and will match what is returned by the appletalktype compatibility operator.

**Legal Values:** Any string of 32 or fewer non-null characters.

**Errors:** typecheck

**EtherTalkZone** string This parameter represents the zone piece of the EtherTalk entity name.

**Legal Values:** Any string of 32 or fewer non-null characters.

**Errors:** typecheck

**HasNames** boolean (Read-only) This constant always returns a value of false. For the general definition of HasNames, see Table 3.4.

**Legal Value:** false

**Errors:** none

**Interpreter** name For the general definition of Interpreter, see Table 3.5.

**Legal values:** /PostScript, /AutoSelect, /Diablo630, /EpsonFX850, /HP7475A, /LaserJetIII, /LaserJetIIP, /ProprinterXL

**Errors:** configurationerror, rangecheck, typecheck

**On** boolean For the general definition of On, see Table 3.5.

**Legal Value:** true, false

**Errors:** configurationerror, typecheck

**Type** name (Read-only) This constant always returns a value of Communications. For the general definition of Type, see Table 3.3 on page 53.

**Legal value:** /Communications

**Errors:** none
### Table 3.9 Parameters typically present in parallel communications parameter sets

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>boolean</td>
<td>For the general definition of <strong>Enabled</strong>, see Table 3.5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal Value:</strong> true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> configurationerror, typecheck</td>
</tr>
<tr>
<td>Handshake</td>
<td>integer</td>
<td>This parameter indicates the hardware/software signal interface that will be used for communications across the parallel (or Centronics) interface. If this key is not present, the default is unidirectional Centronics.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 unidirectional communications commonly used by PCs and PC-compatibles</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 bidirectional communications as specified by version 0.6 of the Hewlett-Packard Boise specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal Values:</strong> 0, 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
<tr>
<td>HasNames</td>
<td>boolean</td>
<td>(Read-only) This constant always returns a value of false. For the general definition of <strong>HasNames</strong>, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal Value:</strong> false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
<tr>
<td>Interpreter</td>
<td>name</td>
<td>For the general definition of <strong>Interpreter</strong>, see Table 3.5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> /PostScript, /AutoSelect, /Diablo630, /EpsonFX850, /HP7475A, /LaserJetIII, /LaserJetIP, /ProprinterXL</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> configurationerror, rangecheck, typecheck</td>
</tr>
<tr>
<td>On</td>
<td>boolean</td>
<td>For the general definition of <strong>On</strong>, see Table 3.5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal Value:</strong> true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> configurationerror, typecheck</td>
</tr>
</tbody>
</table>
**OutputDevice** string  This parameter specifies which communications device is to be used for `stdout` and `stderr`. If the value of **OutputDevice** is the empty string, `stdout` and `stderr` information will be forfeited.

**Legal Values:**  `%Serial%`, `%SerialB%`, `%SerialC%`, etc., `%Parallel%`, `%ParallelB%`, `%ParallelC%`, etc., the empty string

**Errors:**  rangecheck

**Protocol** name  For the general definition of **Protocol**, see Table 3.6.

**Legal values:**  `/Binary`, `/Normal`, `/Raw`, `/TBCP`

**Errors:**  configurationerror, rangecheck, typecheck

**Type**  *(Read-only)* This constant always returns a value of `/Communications`. For the general definition of **Type**, see Table 3.3 on page 53.

**Legal values:**  `/Communications`

**Errors:**  none

---

**Table 3.10 Parameters typically present in SCSI communications parameter sets**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enabled</strong></td>
<td>boolean</td>
<td>For the general definition of <strong>Enabled</strong>, see Table 3.5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal Value:</strong>  <code>true, false</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong>  configurationerror, typecheck</td>
</tr>
<tr>
<td><strong>HasNames</strong></td>
<td>boolean</td>
<td><em>(Read-only)</em> This constant always returns a value of <code>false</code>. For the general definition of <strong>HasNames</strong>, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal Value:</strong>  <code>false</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong>  none</td>
</tr>
<tr>
<td><strong>Interpreter</strong></td>
<td>name</td>
<td>For the general definition of <strong>Interpreter</strong>, see Table 3.5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong>  <code>/PostScript</code>, <code>/AutoSelect</code>, <code>/Diablo630</code>, <code>/EpsonFX850</code>, <code>/HP7475A</code>, <code>/LaserJetIII</code>, <code>/LaserJetIIP</code>, <code>/ProprinterXL</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong>  configurationerror, rangecheck, typecheck</td>
</tr>
</tbody>
</table>
On

boolean

For the general definition of On, see Table 3.5.

Legal Value: true, false

Errors: configurationerror, typecheck

Type

(Read-only) This constant always returns a value of /Communications. For the general definition of Type, see Table 3.3 on page 53.

Legal values: /Communications

Errors: none

3.5.4 Disk and Cartridge Parameter Tables

Table 3.11 contains a list of the current disk and cartridge parameters. Read-only refers to their access by language operators (e.g., setdevparams, currentdevparams). A read-only parameter can change value but not as the result of operations at the language level. Changes to parameters of type /FileSystem take place immediately.

In Table 3.11, it should be understood that a page is a unit of storage whose size is file-system dependent.

Table 3.11 Parameters typically present in disk (/FileSystem) devices

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockSize</td>
<td>integer</td>
<td>(Read-only) This parameter indicates the disk/cartridge formatting size of a page (for the logical and physical size of the media). The formatting size of a page for a cartridge is 1 byte per block. The formatting size of a page for a disk using the Adobe File System is 1024 bytes per block.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: Any non-zero positive integer (typically 1024).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>Bus</td>
<td>string</td>
<td>(Read-only) With Adobe’s storage device implementation, Bus will indicate the name of the SCSI bus on which this disk resides. This parameter is in the form of a string that can be used as input to setdevparams or currentdevparams to get bus parameters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: %Scsi%, %ScsiB%, %ScsiC%, etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
</tbody>
</table>
### Free

**integer** *(Read-only)* This parameter indicates the amount of free space available on the media for the device (in *pages* were the *page size* is indicated by the parameter `BlockSize`). This parameter is valid only if the device is mounted (that is, `Mounted` is set to `true`). A value of 0 indicates that either the device is not mounted or the media is completely full.

**Legal values:** Any non-negative integer or 0.

**Errors:** none

### HasNames

**boolean** *(Read-only)* This parameter indicates whether the device supports named files. This parameter is valid only if the device is mounted (that is, `Mounted` is set to `true`). If the device is not mounted, this parameter has a value of `false`.

**Legal values:** true, false

**Errors:** none

### InitializeAction

**integer** This parameter specifies an action for initializing the device. The following are valid values for disks:

- 0 indicates no action and is the value returned when the parameter is read.
- 1 indicates that the current file system (if any) should be deleted and a new one of size `LogicalSize` created (the media is assumed to have been formatted already). The device must first be mounted; otherwise an `ioerror` will result. For more information, see `LogicalSize`.
- 2 reformats the entire media before creating a new file system of size `LogicalSize`. The `Interleave` parameter also plays a role in how the media is to be formatted. See `Interleave` below for details.
- 3 or greater has the same effect as the value 2 and also carries out product dependent actions, which typically consist of reformatting the disk and running integrity tests before creating the file system. Some devices can have additional parameters that serve as arguments to `InitializeAction`.

**Legal values:** Any non-negative integer.

**Errors:** none

### Interleave

**integer** The purpose of `Interleave` is to arrange logically contiguous sectors on the disk in a way that is most efficient for the system using that disk. This parameter is used only when the media is being formatted (refer to `InitializeAction`, described above).
For example, assume there are 16 sectors going around a single track on a disk. If the first sector has a logical number of 1, the second 2, the third 3, etc., it is referred to as “1 to 1 interleave” and the value of /Interleave is 1. In this case the system must be very fast in order to be able to take data from the disk, one sector immediately after another. If the system fails to consume the first sector in time for the second sector, the system will have to wait an entire revolution of the disk to get the next sector. This can give very poor performance.

If the first sector has a logical number of 1, the third has a logical number of 2, the fifth has a logical number of 3, etc., the system will need to be able to consume the current sector while the head passes over a sector in time for the next logical sector. This is referred to as “2 to 1 interleave” and the value of /Interleave is 2. The sectors in between are used for higher logical numbers and it takes a minimum of two revolutions to get an entire track’s data off the disk. In this example, the second physical sector on the disk would be between logical sectors 1 and 2 and it would be logical number 9.

Similarly, “3 to 1 interleave” has an /Interleave value of 3 and the first sector has a logical number of 1, the fourth one of 2, etc. Normally, the interleave should be set to a value that allows the software to use the information during the time between sectors, but not waste any time. It is difficult to determine what the proper value is and it is highly dependent on the job accessing the disk. Some drives provide buffering for a full track of data. For these drives, “1 to 1 interleave” is almost always most efficient.

**Legal values:** Any positive integer; the legality of the value is disk-dependent.

**Errors:** ioerror

**LogicalSize** integer

When set, this parameter specifies the size of the file system to be created and is used as an argument to the action carried out by InitializeAction. If LogicalSize is 0, InitializeAction uses a default size that is normally the size of the entire media within the device. For more information, see InitializeAction.

When queried, this parameter indicates the current size of the file system on the device (in pages were the page size is indicated by the parameter Block-Size). A value of 0 indicates that the device is not mounted.
If `LogicalSize` is set with a certain value and then the device is reformatted, a query of `LogicalSize` should return the value that was set. However, if the parameter is queried at any time before the media within the device is reformatted, it may return a different value from what was set because it may return the *current size*.

**Legal values:** Any non-negative integer or 0. The value of `LogicalSize` must be less than or equal to the value of `PhysicalSize`.

**Errors:** rangecheck, typecheck

- **Mounted** boolean

  If this parameter is set to `true`, the system attempts to mount the device. If set to `false`, the system attempts 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 if it does not contain a valid file system.

  When queried, the return value indicates whether the device is *currently* mounted. Obtain the result of an attempted mount by querying `Mounted` immediately after setting it.

  `Mounted` raises a **configurationerror** if it is set to `true` and mounting fails, or if it is set to `false` and dismounting fails.

  **Legal values:** `true`, `false`

  **Errors:** configurationerror, typecheck

- **PhysicalSize** integer

  *(Read-only)* This parameter indicates the size of the media (in *pages* were the page size is indicated by the parameter `BlockSize`). This parameter is only valid when the device is mounted (that is, `Mounted` is set to `true`). A value of 0 indicates that the device is not mounted.

  **Legal values:** Any non-negative integer or 0.

  **Errors:** none

- **Removable** boolean

  *(Read-only)* This parameter indicates whether the device supports removable media.

  **Legal values:** `true`, `false`

  **Errors:** none

- **Searchable** boolean

  This parameter indicates whether the device participates in searches in file system operations that have specified a file name without specifying a device. See section 3.8.2, “Named Files” of the *PostScript Language Reference Manual, Second Edition* for more information.
Note  Devices that support removable media (on some products) will initially have Searchable be false. Searchable should be explicitly set to true when it is desired to have the media be searched.

**Legal values:**  true, false

**Errors:**  none

**SearchOrder**  integer  This parameter indicates the priority at which the device participates when searching for a file in operations where no device has been specified. A lower integer indicates a higher priority. This parameter is ignored if the Searchable parameter is false.

**Legal values:**  Any non-negative integer.

**Errors:**  none

**Type**  name  *(Read-only)* This constant always returns a value of FileSystem. For the general definition of Type, see Table 3.3 on page 53.

**Legal values:**  /FileSystem

**Errors:**  none

**Writeable**  boolean  *(Writeable, but only during a mount)* This parameter indicates whether the files on the device can be open for write access. This parameter can be set to true or false only during a mount (that is, when Mounted is being set to true in a call to setdevparams) and only if the media is not write-protected. If the media is already write-protected, this parameter will be a constant equal to false. When the device is not mounted, this parameter indicates whether or not the drive will support writeable media.

**Legal values:**  true, false

**Errors:**  none

---

**Table 3.12  Parameters typically present in cartridge file system (/FileSystem) devices**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockSize</td>
<td>integer</td>
<td><em>(Read-only)</em> For the general definition of BlockSize, see Table 3.11.</td>
</tr>
</tbody>
</table>

**Legal values:**  Any non-zero positive integer (typically 1).

**Errors:**  none
### CartridgeID  
**Type:** integer  
*(Read-only)* This parameter indicates an ID that uniquely identifies this cartridge on a product. CartridgeID is used by the interpreter to determine if a cartridge has been removed from a slot and a different cartridge inserted.

**Legal values:** Any integer.  
**Errors:** none

### CartridgeType  
**Type:** integer  
*(Read-only)* This parameter indicates the category classification of the cartridge. This classification is a registry maintained by Adobe Systems.

**Legal values:** Any integer.  
**Errors:** none

### Free  
**Type:** integer  
*(Read-only)* For the general definition of Free, see Table 3.11.

**Legal values:** Any non-negative integer or 0.  
**Errors:** none

### HasNames  
**Type:** boolean  
*(Read-only)* For the general definition of HasNames, see Table 3.11.

**Legal values:** true, false  
**Errors:** none

### InitializeAction  
**Type:** integer  
This parameter specifies an action for initializing the device. The following are valid values for writeable cartridges (setting InitializeAction for a read only cartridge has no effect):

- 0 indicates no action and is the value returned when the parameter is read.
- 1 reformats the entire media and then creates a new file system using the full size of the cartridge.

**Legal values:** 0 or 1.  
**Errors:** none

### LogicalSize  
**Type:** integer  
For the general definition of LogicalSize, see Table 3.11.

**Legal values:** Any non-negative integer or 0. The value of LogicalSize must be less than or equal to the value of PhysicalSize.  
**Errors:** rangecheck, typecheck
Mounted  boolean  For the general definition of Mounted, see Table 3.11.

Legal values:  true, false

Errors:  configurationerror, typecheck

PhysicalSize  integer  (Read-only) For the general definition of PhysicalSize, see Table 3.11.

Legal values:  Any non-negative integer or 0.

Errors:  none

Removable  boolean  (Read-only) For the general definition of Removable, see Table 3.11.

Legal values:  true, false

Errors:  none

Searchable  boolean  For the general definition of Searchable, see Table 3.11.

Legal values:  true, false

Errors:  none

SearchOrder  integer  For the general definition of SearchOrder, see Table 3.11.

Legal values:  Any non-negative integer.

Errors:  none

Type  name  (Read-only) This constant always returns a value of FileSystem. For the general definition of Type, see Table 3.3 on page 53.

Legal values:  /FileSystem

Errors:  none

Writeable  boolean  (Writable, but only during a mount) For the general definition of Writeable, see Table 3.11.

Legal values:  true, false

Errors:  none

3.5.5  %os% Device Parameters

The %os% device parameter set is only present in Display PostScript products or in other products which are UNIX-based. It is not present in printers.
### Table 3.13 Parameters typically present in %os% device parameter sets

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasNames</td>
<td>boolean</td>
<td><em>(Read-only)</em> This parameter always returns a value of <em>true</em>. For the general definition of HasNames, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal value: <em>true</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>InitializeAction</td>
<td>integer</td>
<td><em>(Read-only)</em> This parameter always returns a value of 0. For the general definition of InitializeAction, see Table 3.11.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal value: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>Mounted</td>
<td>boolean</td>
<td><em>(Read-only)</em> This parameter always returns a value of <em>true</em>. For the general definition of Mounted, see Table 3.11.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal value: <em>true</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: typecheck</td>
</tr>
<tr>
<td>Removable</td>
<td>boolean</td>
<td><em>(Read-only)</em> This parameter always returns a value of <em>false</em>. For the general definition of Removable, see Table 3.11.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: <em>false</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>Searchable</td>
<td>boolean</td>
<td><em>(Read-only)</em> This parameter always returns a value of <em>true</em>. For the general definition of Searchable, see Table 3.11.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: <em>true</em>, <em>false</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>SearchOrder</td>
<td>integer</td>
<td>This parameter <em>initially</em> returns a value of 2. For the general definition of SearchOrder, see Table 3.11.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: Any non-negative integer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
</tbody>
</table>
Type  name  *(Read-only)* This parameter always returns a value of FileSystem. For the general definition of Type, see Table 3.3 on page 53.

Legal value:  /FileSystem

Errors:  none

Writeable  boolean  *(Read-only)* This parameter always returns a value of true. For the general definition of Writeable, see Table 3.11.

Legal value:  true

Errors:  none

3.5.6 Scsi Bus Parameter Set

The parameters in Table 3.14 relate to the SCSI bus as a whole. The (%Scsi%) parameter set is always present in a printer that has a SCSI bus, even if no devices are present on the bus. If more than one SCSI bus is present, the first is called %Scsi%, the second %ScsiB%, the third %ScsiC%, etc. Changes to SCSI parameters do not take affect until the next time the system is initialized.

Table 3.14  Parameters typically present in %Scsi% parameter sets

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>BootDelay</td>
<td>integer</td>
<td>This parameter indicates how long the disk I/O driver should wait (in seconds) during system initialization for the disk 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. You should set this parameter in accordance with the characteristics of the disk attached to the printer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal Values:</strong> Any positive integer or 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
<tr>
<td>CheckParity</td>
<td>boolean</td>
<td>This parameter indicates if parity on the SCSI bus is to be checked. The default value is usually true.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Warning</strong> Setting CheckParity to true on products that do not support parity checking would be unwise. Refer to the individual product addendum for information on whether or not a given product can do this checking.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal Values:</strong> true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
</tbody>
</table>
**InitiatorID**  
integer  
This parameter is the address on the SCSI bus used by the printer when it serves as initiator. The default value is usually 7.

**Legal Values:** A non-negative integer ranging from 0 to 7.

**Errors:** configurationerror

**Poll**  
integer  
This parameter is a bit-encoded specification of which addresses on the SCSI bus should be polled by the printer when it looks for disks during system initialization. For example, a 1 in bit 0 means poll for disk0. Any bits in this mask which correspond to addresses that are used as the printer’s InitiatorID or TargetID, as the InitiatorID for other hosts on the bus, or as the TargetID of peripherals belonging to other hosts on the bus should be set to 0 (meaning “do not poll”). If the bit is set to poll the address corresponding to the printer’s InitiatorID or Target ID, a configurationerror will be generated. If the bit is set to poll an address that shouldn’t be polled, anomalies may occur on the bus. Poll is expressed as an integer bit mask ranging from 0 to 254 (never 255 since all bits cannot be on — one bit must be reserved for InitiatorID). The default value is usually 127 (7F in hexadecimal).

**Legal Values:** An integer bit mask ranging from 0 to 254.

**Errors:** configurationerror

**TargetID**  
integer  
This parameter is the SCSI bus address reserved by the printer for use as the %ScsiComm% communication channel. This address may be the same as the InitiatorID.

**Legal Values:** A positive integer ranging from 0 to 7.

**Errors:** configurationerror

**Type**  
(Read only) This constant always returns a value of /Parameters. For the general definition of Type, see Table 3.3 on page 53.

**Legal values:** /Parameters

**Errors:** none

---

### 3.5.7 Engine Device Parameters

Table 3.15 lists the parameters associated with the %Engine% device. Some products may also have an %Engine_NV% set.
Table 3.15  Parameters typically present in %Engine% device parameter sets

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSizeStandard</td>
<td>name</td>
<td>This parameter assists the engine in determining the physical dimensions of the paper when B4 or B5 paper is selected. There are two choices for the value of BSizeStandard:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ISO: ISO is the abbreviation for the International Standards Organization, which is the international body that defines the “metric” paper sizes (A4, A3, B5, B4, B3, etc.). These are the paper sizes used in Europe and much of the rest of the world. The table below lists the dimensions for the B4 and B5 paper sizes as defined by ISO:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Paper Size</strong></td>
</tr>
<tr>
<td>B4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>JIS: JIS is the abbreviation for the Japanese Institute for Standardization, the national body which specifies standards for use in the country of Japan. Japan also uses the standard “A” paper sizes. However, they use a slightly different definition of the “B” paper sizes. The table below lists the dimensions for the B4 and B5 paper sizes for JIS:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Paper Size</strong></td>
</tr>
<tr>
<td>B4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> In the above tables, a “default unit” denotes 1/72 of an inch.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> ISO, JIS</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> rangecheck</td>
</tr>
<tr>
<td>Darkness</td>
<td>real</td>
<td>This parameter controls the overall lightness or darkness of the rendered page. This parameter does not affect the frame buffer, nor does it have any computational overhead. Legal values are real numbers from 0.0 through 1.0. A value of 0.0 means minimum darkness, 1.0 means maximum darkness. This option is provided in some products whose marking hardware allows software control of colorant application. The default value is product dependent.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> Real numbers in the range [0.0 –1.0] inclusive.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> rangecheck</td>
</tr>
</tbody>
</table>
PageCount  integer  This parameter is a count of all pages fed by the engine. The count includes all of the pages successfully printed as well as the pages that were jammed or spoiled. The value of \texttt{PageCount} is determined by querying the engine.

\textbf{Legal values:} Any non-negative integer or 0.

\textbf{Errors:} typecheck

TimeToStandby  integer  After the specified number of minutes, the 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 acceptable values for \texttt{TimeToStandby} are product-specific. An unallowed value will be rounded to the nearest allowed value.

Specifying a value of 0 for this parameter has the affect of never letting the printer enter the “standby” mode.

\textbf{Legal values:} Product-specific, although typically an integer in the range \([0 – 720]\) inclusive.

\textbf{Errors:} none

Type  name  (Read-only) This constant always returns a value of Parameters. For the general definition of \texttt{Type}, see Table 3.3 on page 53.

\textbf{Legal value:} /Parameters

\textbf{Errors:} none

### 3.5.8 Emulator Parameters

An emulator is an alternative interpreter for the input stream. Some PostScript printers have the ability to emulate other printers. The \texttt{Interpreter} device parameter (described in Table 3.5) specifies what rules a printer will use to interpret the stream of input characters in order to make marks on the page. If the value of the \texttt{Interpreter} parameter is something other than /PostScript, the printer is being asked to emulate the functionality of some other printer.

For example, the Diablo630 is a daisy wheel printer which has very limited capabilities other than putting marks on a page. The input stream is code for characters; the printer assumes one character to follow another until a carriage return or line feed is reached.

Thus, to emulate a Diablo630 printer, the code:

\[
(%\text{SerialA}%)<</\text{Interpreter} /\text{Diablo630} \\
/\text{Protocol} /\text{Raw}> setdevparams
\]
gives Diablo630-like functionality to input channel SerialA on a PostScript printer that has a Diablo630 emulator. This functionality will be invoked at the next job boundary.

The LaserJetIII emulator, the LaserJetIIP emulator, the color version of the HP7475A plotter emulator, and the Diablo630 emulator have parameters which allow the user to specify default values. The emulator parameters can be set with the `setdevparams` operator and read with the `currentdevparams` operator.

Tables 3.16 through 3.19 describe the parameters for the LaserJetIII emulator, the LaserJetIIP emulator, the color version of the HP7475A plotter emulator, and the Diablo630 emulator.

| Table 3.16 Parameters typically present for the LaserJetIII emulator |
|------------------------|------------------------|

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplex</td>
<td>integer</td>
<td>This parameter sets the initial state of duplexing within a PCL job for printers which are capable of duplex operation. Language commands within the print stream can override the setting of this parameter. Acceptable values for <strong>Duplex</strong> are listed below.</td>
</tr>
<tr>
<td>Value</td>
<td>Meaning</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>simplex</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>long-edge binding duplex</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>short-edge binding duplex</td>
<td></td>
</tr>
</tbody>
</table>

The default value for **Duplex** is 0, which means that duplexing is not performed.

<table>
<thead>
<tr>
<th>Landscape</th>
<th>boolean</th>
<th>If true, the default orientation of the page will be landscape unless otherwise specified in the PCL of the page.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>LineWrap</th>
<th>boolean</th>
<th>If true, long lines wrap to the next line. If false, long lines are truncated.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>MaxLJMemory</th>
<th>integer</th>
<th>This parameter specifies the maximum amount of memory that the emulator will ask for from the page allocator to store downloaded fonts and macros. This limit is important because the emulator will acquire memory at the expense of the PostScript interpreter’s memory needs, such as VM or the font cache. <strong>MaxLJMemory</strong> is rounded to the nearest multiple of a memory block size (8192 bytes).</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PaperSize</th>
<th>integer</th>
<th>This parameter sets the paper size to be used within the PCL job. This parameter has results similar to the “paper size command” ([Esc]&amp;l#A) within the PCL5 language.</th>
</tr>
</thead>
</table>
The `PaperSize` parameter can specify any of the supported page sizes available to the LaserJetIII printer. In addition, there is a special value, −1, which means “unspecified.” This allows the printer to draw paper from the default slot. The paper sizes available to the LaserJetIII printer and their associated integer values are listed below.

<table>
<thead>
<tr>
<th>Value</th>
<th>Paper Size</th>
<th>Dimensions (in default units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>−1</td>
<td>unspecified</td>
<td>default slot</td>
</tr>
<tr>
<td>1</td>
<td>Executive</td>
<td>522 x 756</td>
</tr>
<tr>
<td>2</td>
<td>Letter</td>
<td>612 x 792</td>
</tr>
<tr>
<td>3</td>
<td>Legal</td>
<td>612 x 1008</td>
</tr>
<tr>
<td>26</td>
<td>A4</td>
<td>595 x 842</td>
</tr>
<tr>
<td>80</td>
<td>Monarch Envelope</td>
<td>279 x 540</td>
</tr>
<tr>
<td>81</td>
<td>Com-10 Envelope</td>
<td>297 x 684</td>
</tr>
<tr>
<td>90</td>
<td>International DL Envelope</td>
<td>312 x 624</td>
</tr>
<tr>
<td>91</td>
<td>International C5 Envelope</td>
<td>459 x 649</td>
</tr>
</tbody>
</table>

*Note* In the above tables, a “default unit” denotes 1/72 of an inch.

The default value of `PaperSize` is −1, indicating “unspecified,” the default tray.

- **WaitTimeout**: integer
  - This parameter specifies the wait timeout (in seconds) after which a page is ejected. The default is 30.

- **TopMargin**: integer
  - Amount of white space at the top of the page, specified in IPU (1/7200 inch). The default is 3600 (1/2 inch).

- **Type**: name
  - `Type` has the value of `Emulator`. For the general definition of `Type`, see Table 3.3 on page 53.

- **VMI**: real
  - This parameter specifies the space between lines of text in 1/7200 inch units.

---

**Table 3.17 Parameters typically present for the LaserJetIII emulator**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copies</td>
<td>integer</td>
<td>This parameter specifies the default number of copies of a document to be printed.</td>
</tr>
<tr>
<td>FontFixed</td>
<td>boolean</td>
<td>If true, a fixed pitch font is requested. If it is false, a proportional spaced font is requested.</td>
</tr>
<tr>
<td>FontHeight</td>
<td>real</td>
<td>This parameter specifies the desired font height in 1/72 of an inch units.</td>
</tr>
<tr>
<td>FontItalic</td>
<td>boolean</td>
<td>If true, an italic (or oblique) font is requested.</td>
</tr>
<tr>
<td>FontPitch</td>
<td>real</td>
<td>This parameter is used only if <code>FontFixed</code> is true. <code>FontPitch</code> takes a real number specifying the number of characters per inch.</td>
</tr>
</tbody>
</table>
### 3.5 Device Parameters

**FontSymbolSet**  
integer  
This parameter specifies the mapping from 7 or 8 bit numbers to glyphs that appear on the page. The value of this parameter is the number associated with this field in a downloaded font.

**FontTypeface**  
integer  
The value of **FontTypeface** is the number assigned to a particular font (for example, Times, Helvetica, Palatino, etc.). The integer value (which can be up to 16 bits) comes from a table published by Hewlett-Packard.

**FontWeight**  
integer  
This parameter specifies the “weight” or “boldness” of desired font. The parameter ranges from -7 to +7 where -7 is very light and +7 is very bold.

**Landscape**  
boolean  
If true, the initial orientation of the page will be landscape instead of portrait.

**LinesPerInch**  
real  
This parameter specifies the default value for the “vertical motion index.” This determines the interline spacing (and hence the number of lines on the page).

**ManualFeed**  
boolean  

**MaxLJMemory**  
integer  
This parameter allows the user to limit the amount of memory that the LaserJetIIP emulator will take for its needs. This limit is important because the emulator will acquire memory at the expense of the PostScript interpreter’s memory needs, such as VM or the font cache. Within a given emulation job, the LaserJetIIP emulator will use temporary memory in excess of **MaxLJMemory** to hold fonts and macros.

**Type**  
name  
**Type** has the value of Emulator. For the general definition of **Type**, see Table 3.3 on page 53.

**WaitTimeout**  
integer  
The value of **WaitTimeout** (in seconds) is used by the LaserJetIIP emulator as the minimum amount of time the emulator will wait for additional incoming characters before declaring the job finished. A value of 0 indicates to the emulator that it should wait forever. The parameter typically has a default value of 30.

---

**Table 3.18**  
Parameters typically present for the color version of the HP7475A plotter emulator

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
</table>
| ColorSetup    | string   | This parameter allows the user to change the default pen color. The **ColorSetup** parameter is a string which contains a list of numbers. There must be a multiple of five numbers in the string. Each set of five specifies the
pen number (integer), width of the pen’s line in millimeters (real), the red color value (real, between 0 and 1.0), the green color value (real, between 0 and 1.0), and the blue color value (real, between 0 and 1.0).

Table 3.19 Parameters typically present for the Diablo630 emulator

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoLF</td>
<td>boolean</td>
<td>If true, automatic line feeding is specified.</td>
</tr>
<tr>
<td>BoldFontName</td>
<td>name</td>
<td>This parameter specifies the name of the PostScript font used for boldface printing when ECS is false.</td>
</tr>
<tr>
<td>ECS</td>
<td>boolean</td>
<td>If true, the printer emulates the IBM PC Graphics ECS (extended character set) print wheel. If false, the printer emulates the 96-character plastic print wheel.</td>
</tr>
<tr>
<td>ECSDataWidth</td>
<td>integer</td>
<td>Selects 7- or 8-bit data when ECS is true.</td>
</tr>
<tr>
<td>RegFontName</td>
<td>name</td>
<td>This parameter specifies the name of the PostScript font used for regular printing when ECS is false.</td>
</tr>
</tbody>
</table>

3.6 The Fax Environment Interface

This section describes the facilities available for setting, controlling and examining the fax environment shared by all jobs and all users connected to the fax printer. These facilities fall into two categories: fax device parameters and administrative resources.

3.6.1 Fax Device Parameters

Several device settings can be established which control various aspects of the facsimile transmission and receipt. These are stored within the fax printer in non-volatile memory. They persist through multiple fax jobs and are consistent across multiple users connected to a fax printer. These device settings are accessed by means of the %Fax% and %Calendar% device parameter sets which are accessed through the currentdevparams and setdevparams operators (see 3.5, “Device Parameters”)

3.6.2 The %Fax% Device

The %Fax% device is used to control the general, global operation of the fax capability of the fax printer. The semantics of the various parameters in the %Fax% device parameter set are described in Table 3.20.
### Table 3.20 Parameters typically present in the `%Fax% device parameter set

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivityReport</td>
<td>boolean</td>
<td>If <code>true</code>, then an activity report will be printed automatically whenever the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>activity buffer is full. If <code>false</code>, then no activity reports will be printed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>automatically. Printing of the report does not clear the recorded information;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the oldest entries are overwritten with new entries. Reports may always be gen-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>erated by request from the host computer (see also section 3.6.4, “Adminis-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>trative Resources.”). The default is <code>false</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> <code>true</code>, <code>false</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> <code>typecheck</code></td>
</tr>
<tr>
<td>DefaultCaptionOn</td>
<td>boolean</td>
<td>This parameter determines whether the default page caption routine will</td>
</tr>
<tr>
<td></td>
<td></td>
<td>place captions on the fax pages. The default value is <code>true</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> <code>true</code>, <code>false</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> <code>typecheck</code></td>
</tr>
<tr>
<td>DefaultConfirmOn</td>
<td>boolean</td>
<td>This parameter determines whether confirmation reports will be produced by</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the default procedures. The default value is <code>true</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> <code>true</code>, <code>false</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> <code>typecheck</code></td>
</tr>
<tr>
<td>DefaultCoversOn</td>
<td>boolean</td>
<td>This parameter determines whether the default page cover procedure should</td>
</tr>
<tr>
<td></td>
<td></td>
<td>actually generate cover pages. The default value is <code>true</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> <code>true</code>, <code>false</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> <code>typecheck</code></td>
</tr>
<tr>
<td>DefaultResolution</td>
<td>integer</td>
<td>If the value of <code>FaxType</code> in the <code>FaxOptions</code> dictionary is <code>null</code>, then the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value of this parameter determines the resolution of the raster fax being</td>
</tr>
<tr>
<td></td>
<td></td>
<td>prepared. It must have a value of <code>0</code> or <code>1</code>; <code>1</code> is the default and specifies</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to use fine CCITT group 3 resolution.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> <code>0</code>, <code>1</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> <code>rangecheck</code>, <code>typecheck</code></td>
</tr>
</tbody>
</table>
**DefaultRetryCount**  
**integer**  
This integer parameter is the number of retries attempted if the value of **MaxRetries** in the options dictionary is null. It must be in the range of 0 to 100 inclusive. The default value is 0.

**Legal values:** Any non-negative integer in the range of 0 to 100 inclusive.

**Errors:** rangecheck, typecheck

**DefaultRetryInterval**  
**integer**  
If **RetryInterval** in the options dictionary has a value of null, this value is used as the number of minutes to wait between retries. It has a default value of 3, and must be in the range of 1 to 60.

**Legal values:** Any positive integer in the range of 1 to 60.

**Errors:** rangecheck, typecheck

**DialToneWaitPeriod**  
**integer**  
At the onset of dialing, and whenever a “w” is encountered in the dialing string, the fax printer waits for a dial tone. This parameter indicates the maximum number of seconds to wait each time. If no dial tone is heard within this period of time, the wait will be considered a failure. The default value is 1.

**Legal values:** Any positive integer in the range of 1 to 10.

**Errors:** rangecheck, typecheck

**ID**  
**string**  
Provides the string by which the fax machine identifies itself to other fax machines. It is usually set to the company name or the telephone number of the line to which the machine is attached. The string can have up to 20 characters and is defined according to the 1988 CCITT fax protocol.

**Legal values:** A string of up to 20 characters.

**Errors:** limitcheck, typecheck

**MaxFaxBuffer**  
**integer**  
This parameter sets an upper bound on the number of bytes of printer RAM which may be used for incoming and outgoing fax data. It is only relevant if **StorageDevice** has a value of (%ram%). This is only an upper limit; no space is set aside based on this parameter’s value. The minimum value for **MaxFaxBuffer** is 350000. The default and maximum values are product specific and may be a function of the actual amount of memory in the fax printer.

**Legal values:** Product dependent

**Errors:** rangecheck, typecheck
**PostScriptPassword**

*string*

This password is used by the receiving machine to determine if the sender is authorized to send PostScript language programs. In certain situations, a particular machine may only want to receive PostScript language programs from certain other machines. For this reason, the **PostScriptPassword** exists. It has a maximum length of 32 characters. Its current value is not returned by the **currentdevparams** operator; a null string is always shown as the value associated with this key.

When a telephone connection is established and the receiving machine is willing to accept PostScript language files (that is, **ReceivePostScript** is *true*), the sending machine is asked to encrypt some arbitrary value (issued by the receiver) using the **PostScriptPassword** from the options dictionary passed to the **faxsendps** operator. The receiver then encrypts the arbitrary value with its own **PostScriptPassword** and compares the encrypted results.

If the encrypted results compare, the PostScript language file transmission can take place. If the encrypted results do not compare, the receiver refuses to accept the PostScript language file transmission from the sender and the telephone connection is broken. Depending upon the value of **RevertToRaster** in the **FaxOptions** dictionary of the sender, the sending machine may or may not revert to a raster image fax transmission at this point.

This password mechanism can be sidestepped if the **PostScriptPassword** value in the receiving machine is the empty string. In this case, all jobs received in PostScript language form will be accepted (if the value of **ReceivePostScript** is *true*). However, when they are run, attempts to change system or device parameters will fail unless accompanied with a password equal to the **SystemParamsPassword**. If the **SystemParamsPassword** is not set, attempts to change system or device parameters will fail. Those jobs will also not be able to generate outbound faxes themselves or to access any of the **FaxAdminOps** facilities.

**Legal values:** A string of up to 32 characters

**Errors:** invalidaccess, limitcheck, typecheck

**ReceivePostScript**

*boolean*

If *true*, the machine is willing to receive and execute fax jobs that are PostScript language files. This is the default. If *false*, then any attempt to send PostScript language files to this machine will be rejected. Only faxes in CCITT compressed raster form will be accepted.

**Legal values:** *true, false*

**Errors:** typecheck
Rings integer

In order to allow a telephone to be shared by the fax printer and a person, the printer needs to let the phone ring several times to give the person time to answer. If the phone is not shared, it is desirable for the printer to answer the phone as quickly as possible. The integer value supplied with this key determines how many rings of the telephone will be ignored before the fax printer answers. For example if the value is 2 the printer will attempt to answer the phone just after the second ring. The minimum value is 1, the maximum 30.

Legal values: Any positive integer in the range of 1 to 30

Errors: rangecheck, typecheck

ServiceEnable integer

This parameter acts as the master on-off control for fax send and receive functionality. The parameter takes an integer value with these meanings:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Fax completely disabled</td>
</tr>
<tr>
<td>1</td>
<td>Send only enabled</td>
</tr>
<tr>
<td>2</td>
<td>Receive only enabled</td>
</tr>
<tr>
<td>3</td>
<td>Both send and receive enabled</td>
</tr>
</tbody>
</table>

The default value is 3. If receive is not enabled, then the ringing phone will not be answered. If send is not enabled, then execution of `faxsendps` or any execution of `setpagedevice` that attempts to establish a Fax OutputDevice will fail and an I/O error will be recorded.

Legal values: 0, 1, 2, or 3

Errors: rangecheck, typecheck

Speaker integer

This parameter controls the use of the speaker associated with the fax modem. The use of the speaker is determined by an integer code:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>off at all times</td>
</tr>
<tr>
<td>1</td>
<td>on until a connection is established, 10-20 seconds</td>
</tr>
<tr>
<td>2</td>
<td>on at all times</td>
</tr>
</tbody>
</table>

The default value is 1 which allows a human to monitor connections being made for both transmissions and receptions.

Legal values: 0, 1, or 2

Errors: rangecheck, typecheck

StorageDevice string

This parameter designates the name of the storage device used to hold received fax data (either PostScript language files or raster) before it is printed, and to hold outgoing fax data (either PostScript language files or raster) before it is transmitted. The default is a value of (%ram%) which
means that printer RAM is to be used. For devices other than %ram% (such as a disk or cartridge), a parameter set must exist and the parameters Mounted and Writeable must be defined and set to true.

If there is a disk on the fax printer, setting this parameter to its name (for example, (%disk0%)) will mean that the disk is used for storage. Because there is typically more space available on disk than in RAM, this will mean that longer faxes can be received and that longer faxes can be transmitted with just one phone call. Changes to this parameter do not take affect until the fax printer is re-initialized.

**Legal values:** String name of any writeable storage device.

**Errors:** ioerror, limitcheck

**Type**

(Read only) This constant always returns a value of /Parameters. For the general definition of Type, see Table 3.3 on page 53.

**Legal values:** /Parameters

**Errors:** none

**WaitForDialTone**

boolean This parameter designates whether the fax printer will insist on hearing a dial tone before it starts to dial. If WaitForDialTone is false, the call will be dialed regardless of whether a dial tone is heard. If WaitForDialTone is true, then the fax printer will listen (for up to DialToneWaitPeriod seconds) for a dial tone. If one is heard, dialing will start. If one is not heard, then the call will be considered a failure similar to a busy signal (that is, the retry count will be decremented, etc.). The default value is true.

**Legal values:** true, false

**Errors:** typecheck

### 3.6.3 The %Calendar% Device

The fax printer has a battery powered time-of-day clock which is used to provide the time for cover sheets and scheduling fax transmissions. This clock must be set once initially, and then twice a year to follow daylight savings time.

The string (%Calendar%) identifies the calendar device and the entries in the dictionary describe the local date and time. The entries in the dictionary are described in Table 3.21.
### Table 3.21  Parameters typically present in the %Calendar% device parameter set

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
</table>
| Day    | integer | This parameter represents the day.  
**Legal Values:** An integer in the range of 1 to 31.  
**Errors:** rangecheck |
| Hour   | integer | This parameter represents the hour.  
**Legal Values:** An integer in the range of 0 to 23.  
**Errors:** rangecheck |
| Minute | integer | This parameter represents the minute.  
**Legal Values:** An integer in the range of 0 to 59.  
**Errors:** rangecheck |
| Month  | integer | This parameter represents the month.  
**Legal Values:** An integer in the range of 1 to 12.  
**Errors:** rangecheck |
| Running| boolean | This parameter turns the clock off and on. When turning the clock on (setting the value to true), the time elements should also be set at the same time in order to avoid a rangecheck error.  
The clock must be on in order to set the time. If the clock is turned off (to preserve battery power) or is assumed to be inaccurate, the time returned will be January 1, 1980 00:00:00.  
**Legal Values:** true, false  
**Errors:** rangecheck |
| Second | integer | This parameter represents the second.  
**Legal Values:** An integer in the range 0 to 59.  
**Errors:** rangecheck |
3.6  The Fax Environment Interface

Year integer This parameter represents the year. The value of this parameter returned by currentdevparams, has special significance. If it is non-zero and in the range 1980 to 2079, then it represents the year. If it is 0, then the clock is either turned off (to preserve battery power) or is assumed to be inaccurate.

Legal Values: An integer in the range 1980 to 2079.

Errors: rangecheck

3.6.4 Administrative Resources

Operators have been defined to assist in overseeing the general operation of the fax printer. Job records and telephone connections can be checked with these operators. The operators are available as a PostScript Level 2 resource named /FaxAdminOps in the category /ProcSet. They may be reached by use of the PostScript Level 2 findresource operator:

/FaxAdminOps /ProcSet findresource begin

In this example, a dictionary containing the administrative fax operator definitions is placed on the operand stack. The begin that follows moves that dictionary to the dictionary stack so that the operators can be executed directly.

Job Records

The fax printer keeps logs on various aspects of all transmitted and received fax jobs. These records can be accessed by using the four operators described in Table 3.22.

Table 3.22 Operators for accessing fax job logs

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proc deletejobsforall</td>
<td>Selectively deletes entries from the logs.</td>
</tr>
<tr>
<td>proc jobsforall</td>
<td>Allows programmers to perform other functions besides reports.</td>
</tr>
<tr>
<td>proc reportjoblist</td>
<td>Prints one or more pages on the printer listing the log entries. This is similar to the activity report which may be generated automatically. For more information, see the %Fax% parameter ActivityReport in Table 3.20.</td>
</tr>
<tr>
<td>proc returnjoblist</td>
<td>Sends an ASCII string to the connected computer made up of individual log entries. The components of each log entry are separated by double colons (::).</td>
</tr>
</tbody>
</table>

3.6  The Fax Environment Interface
Each of these operators goes through the list of log entries. For each entry, a job dictionary is placed on the operand stack, and then the procedure proc is executed. proc is expected to consume the job dictionary from the operand stack. For deletejobsforall, the proc must leave a boolean on the stack indicating whether the log entry should be deleted – true means delete. For reportjoblist and returnjoblist, the proc must leave a boolean on the stack indicating whether the entry is to be included in the report – true means it will be included. For jobsforall the proc should carry out whatever actions it desires and not push a return value on the stack.

Table 3.23 Entries in a job dictionary

<table>
<thead>
<tr>
<th>Entry</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallCount</td>
<td>integer</td>
<td>The value of CallCount indicates the number of calls involved in this transmission; it is also the number of entries in the Calls array. This value is always 1 for received faxes.</td>
</tr>
<tr>
<td>Calls</td>
<td>array of dictionaries</td>
<td>The dictionaries in the Calls array describe the individual calls of the possibly multiple call fax job. The contents of each dictionary in Calls are described in Table 3.24.</td>
</tr>
<tr>
<td>DialCallee</td>
<td>string</td>
<td>See the description of DialCalle in Table 2.3.</td>
</tr>
<tr>
<td>ErrorArray</td>
<td>array of strings</td>
<td>This is an array of strings which describe status conditions. It is indexed by the ErrorIndex entries described in Table 3.24.</td>
</tr>
<tr>
<td>JobId</td>
<td>integer</td>
<td>This entry is a unique identifier for this send or receive job.</td>
</tr>
<tr>
<td>RecipientName</td>
<td>string</td>
<td>For transmitted faxes, the value of RecipientName is the same as was given in the job’s FaxOptions dictionary. If RecipientName in the job’s FaxOptions dictionary is null, a non-null value to store in the log will be sought according to the scheme described under RecipientName in Table 2.3. For received faxes, this item is not present.</td>
</tr>
<tr>
<td>RecipientOrg</td>
<td>string</td>
<td>For transmitted faxes, the value of RecipientOrg is the same as was given in the job’s FaxOptions dictionary. If RecipientOrg in the job’s FaxOptions dictionary is null, a non-null value to store in the log will be sought according to the scheme described under RecipientOrg in Table 2.3. For received faxes, this item is not present.</td>
</tr>
<tr>
<td>RecipientPhone</td>
<td>string</td>
<td>For transmitted faxes, the value of RecipientPhone is the same as was given in the job’s FaxOptions dictionary. If RecipientPhone in the job’s FaxOptions dictionary is null, a non-null value to store in the log will be sought according to the scheme described under RecipientPhone in Table 2.3.</td>
</tr>
</tbody>
</table>
For received faxes, this item is not present.

**SenderName** string For transmitted faxes, the value of **SenderName** is the same as was given in the job’s **FaxOptions** dictionary. If **SenderName** in the job’s **FaxOptions** dictionary is null, a non-null value to store in the log will be sought according to the scheme described under **SenderName** in Table 2.3.

For received faxes, this item is not present.

**SenderOrg** string For transmitted faxes, the value of **SenderOrg** is the same as was given in the job’s **FaxOptions** dictionary. If **SenderOrg** in the job’s **FaxOptions** dictionary is null, a non-null value to store in the log will be sought according to the scheme described under **SenderOrg** in Table 2.3.

For received faxes, this item is not present.

**TimeBegan** array of integers This parameter indicates the time when the job was submitted.

**TotalPages** integer This parameter indicates the sum of all pages imaged or received.

**TotalPagesSent** integer This parameter indicates the number of pages transmitted (not including cover sheets) or printed.

Table 3.24 describes the entries in a **Calls** dictionary.

**Table 3.24 Entries in a Calls dictionary**

<table>
<thead>
<tr>
<th>Entry</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CallLength</strong></td>
<td>integer</td>
<td>This key designates the length of the call in seconds.</td>
</tr>
<tr>
<td><strong>CoverPagesSent</strong></td>
<td>integer</td>
<td>This key designates a count of the number of cover sheets transmitted in this call; for received faxes this is always 0.</td>
</tr>
<tr>
<td><strong>ErrorIndex</strong></td>
<td>integer</td>
<td>This key designates the final status for the call. This integer can be used as an index into <strong>ErrorArray</strong>.</td>
</tr>
<tr>
<td><strong>FaxKind</strong></td>
<td>integer</td>
<td>This key designates the type of fax transmitted or received. The fax type is determined by an integer code:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>group 3 raster transmission</td>
</tr>
<tr>
<td>1</td>
<td>PostScript language file transmission</td>
</tr>
<tr>
<td>2</td>
<td>group 3 raster reception</td>
</tr>
<tr>
<td>3</td>
<td>PostScript language file reception</td>
</tr>
</tbody>
</table>
Pages integer This key designates the number of pages prepared for this call or received. Compare with the definition of PagesSent.

PagesSent integer This key designates the number of pages actually transmitted or printed. Compare with the definition of Pages.

TimeSent array of integers This key designates the time when the call started.

---

**tryphone**

The *tryphone* operator allows diagnosis of the attachment to the telephone system. The operator takes two arguments and returns a string:

```
string int tryphone string
```

The *string* object is the telephone number to be dialed in the test. The telephone number must be in the same format as described under *DialCallee* in Table 2.3. The input integer *int* is a code indicating the kind of test to be performed. If *int* is 0, *tryphone* will wait until a dial tone is detected after dialing the string. If *int* is 1, *tryphone* will wait until it receives a handshake from the remote fax machine. The returned *string* indicates the results of the test. If *string* is “Heard fax machine.”, then the handshake with the remote fax machine was successful (and the phone connection was dropped politely with no data transmission). Other *strings* indicate the results of the test. The strings that may be returned are listed below.

```
“Heard fax machine.”
“Heard dial tone.”
“No dial tone.”
```

The *tryphone* operator will return an I/O error if fax transmission is not enabled (see *ServiceEnable* in Table 3.20) or if the fax hardware is not properly installed.
In Level 2, PostScript objects such as fonts, patterns, filters, and so on, can be managed as open-ended collections of resources grouped into categories. A resource is requested by giving the resource category and name. If the resource does not reside in VM, the resource management mechanism loads it from an external source, such as a disk, a ROM cartridge, or a network file server. Named resources are discussed in section 3.9, “Named Resources” in the PostScript Language Reference Manual, Second Edition.

The resources listed in Tables 4.1, 4.2, and 4.3 are typically present in all implementations of PostScript language version 2012.

### 4.1 Regular Resources

Regular resources, listed in Table 4.1, are resources whose instances are ordinary, useful objects, such as font or halftone dictionaries.

#### Table 4.1 Regular resources

<table>
<thead>
<tr>
<th>Category name</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorRendering</td>
<td>No instances defined.</td>
</tr>
<tr>
<td>ColorSpace</td>
<td>No instances defined.</td>
</tr>
<tr>
<td>Emulator</td>
<td>No instances defined.</td>
</tr>
<tr>
<td>Encoding</td>
<td>ISOLatin1Encoding StandardEncoding</td>
</tr>
</tbody>
</table>
4.2 Implicit Resources

Implicit resources, listed in Table 4.2, are resources whose instances are not objects, but which represent some built-in capability of the PostScript interpreter.

Table 4.2 Resources whose instances are implicit

<table>
<thead>
<tr>
<th>Category name</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorRenderingType</td>
<td>1</td>
</tr>
<tr>
<td>ColorSpaceFamily</td>
<td>CIEBasedA CIEBasedABC DeviceCMYK DeviceGray DeviceRGB Indexed Pattern Separation</td>
</tr>
<tr>
<td>Filter</td>
<td>ASCII85Decode ASCII85Encode ASCIIHexDecode ASCIIHexEncode CCITTFaxDecode CCITTFaxEncode DCTDecode DCTEncode LZWDecode LZWEncode NullEncode RunLengthDecode RunLengthEncode SubFileDecode</td>
</tr>
<tr>
<td>FMapType</td>
<td>2, 3, 4, 5, 6, 7, 8</td>
</tr>
<tr>
<td>FontType</td>
<td>0, 1, 3, 4, 5</td>
</tr>
<tr>
<td>FormType</td>
<td>1</td>
</tr>
<tr>
<td>HalftoneType</td>
<td>1, 2, 3, 4, 5, 6</td>
</tr>
<tr>
<td>HWOptions</td>
<td>Clock (TODClock) Fax (USModem) PixelBurst Type1Coprocessor</td>
</tr>
<tr>
<td>ImageType</td>
<td>1</td>
</tr>
<tr>
<td>PatternType</td>
<td>1</td>
</tr>
<tr>
<td>ProcSet</td>
<td>FaxOps FaxAdminOps</td>
</tr>
</tbody>
</table>
4.3 Resources Used to Define New Resources

Resources used in defining new resources, listed in Table 4.3, can be used to create new resource categories, each containing an independent collection of named instances. This is accomplished through a level of recursion in the resource machinery itself.

Table 4.3 Resources used in defining new resource categories

<table>
<thead>
<tr>
<th>Category name</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorSpace</td>
<td>ColorSpaceFamily Emulator</td>
</tr>
<tr>
<td>Encoding</td>
<td>Filter FMapType</td>
</tr>
<tr>
<td>Font</td>
<td>FontType Form</td>
</tr>
<tr>
<td>FormType</td>
<td>Generic Halftone</td>
</tr>
<tr>
<td>HalftoneType</td>
<td>ImageType IODevice</td>
</tr>
<tr>
<td>OutputDevice</td>
<td>Pattern PatternType</td>
</tr>
<tr>
<td>ProcSet</td>
<td></td>
</tr>
<tr>
<td>Generic</td>
<td>No instances defined.</td>
</tr>
</tbody>
</table>

4.3.1 Accessing Product Page

Device Capability Information

The resource category **OutputDevice** has been added to perform the following tasks.

- Enable applications to query printer capabilities directly.

- Maintain functional equivalence with Level 1 (where page size capability information was present through enumeration of `letter`, `legal`, `a4`, etc. keys in `userdict`).

The resource category **OutputDevice** is present in interpreters starting with version 2011. This category contains one instance for each **OutputDevice** value which `setpagedevice` can accept for that product. Products which do not contain the **OutputDevice** page device key, that is, which have only one possible page device output device, have the single instance **Default** for the **OutputDevice** category.

The value of each instance of the **OutputDevice** category is a dictionary which contains key-value pairs describing, certain capabilities of that particular output device, such as the possible page sizes or the possible resolutions. This dictionary does not represent the current state of the printer; it simply provides a static list of the possible capabilities of the printer. Over
time, Adobe is likely to define new entries in this dictionary to reflect added capabilities. In 2012 products, the entries listed in Table 4.4 are typically present:

**Table 4.4 Description of keys present in an instance of the category **

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HWResolution</strong></td>
<td>Array of <strong>HWResolution</strong> values which can be supported by the product.</td>
</tr>
<tr>
<td></td>
<td>Each element can be either an array of two numbers indicating a discrete</td>
</tr>
<tr>
<td></td>
<td><strong>HWResolution</strong> support or an array of four numbers ([x1 \ y1 \ x2 \ y2]) indicating that the range of HWResolutions between ([x1 \ y1]) and ([x2 \ y2]) is supported. Redundant values may be present.</td>
</tr>
<tr>
<td><strong>ManualSize</strong></td>
<td>Array of page sizes for the product which can be fed manually.</td>
</tr>
<tr>
<td></td>
<td>Each element can be either an array of two numbers indicating a discrete</td>
</tr>
<tr>
<td></td>
<td>page size supported or an array of four numbers ([x1 \ y1 \ x2 \ y2]) indicating that the range of page sizes between ([x1 \ y1]) and ([x2 \ y2]) is supported. Redundant values may be present. In a product that does not support the <strong>ManualFeed</strong> page device parameter, the array of page sizes should have no entries.</td>
</tr>
<tr>
<td><strong>PageSize</strong></td>
<td>Array of page sizes for the product which can be fed automatically (assuming appropriate media are installed).</td>
</tr>
<tr>
<td></td>
<td>Each element can be either an array of two numbers indicating a discrete</td>
</tr>
<tr>
<td></td>
<td>page size supported or an array of four numbers ([x1 \ y1 \ x2 \ y2]) indicating that the range of page sizes between ([x1 \ y1]) and ([x2 \ y2]) is supported. Redundant values may be present.</td>
</tr>
</tbody>
</table>
CHAPTER 5

Other Extensions to PostScript Level 2

Since the publishing of the Postscript Language Reference Manual, Second Edition, there have been and potentially will continue to be extensions to the Level 2 PostScript Language. This chapter lists those extensions for version 2012.

5.1 Type 6 Halftone Dictionary

This section describes the type 6 halftone dictionary. For more information about the concepts and terms used below, see “Halftones,” section 6.4 of the PostScript Language Reference Manual, Second Edition.

The type 6 halftone dictionary defines a halftone screen directly by specifying a threshold array at device resolution. This is similar to a type 3 halftone dictionary, but the threshold array is obtained from a file instead of a string object. This allows threshold arrays larger than 65535 bytes (the implementation limit for strings); smaller threshold arrays can also be defined this way.

When presented with a type 6 halftone dictionary, sethalftone reads width x height characters from the Thresholds file and saves the resulting threshold array in internal storage. The file must supply sufficient data; if it ends prematurely, a rangecheck error is raised.

When the current halftone is a type 6 halftone dictionary, currenthalftone returns a halftone dictionary whose Thresholds file can be used to access the contents of the current threshold array just as if it were a read-only file. (That is, the Thresholds file object returned by currenthalftone is different from the one that was given to sethalftone.) This file treats the contents of the current threshold array as a circular buffer that can be read repeatedly; it will never reach end-of-file.

Table 5.1 lists the entries in a type 6 halftone dictionary.
### Table 5.1 Entries in a type 6 halftone dictionary

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>HalftoneType</td>
<td>integer</td>
<td><em>(Required)</em> Must be 6.</td>
</tr>
<tr>
<td>Height</td>
<td>integer</td>
<td><em>(Required)</em> Height of the threshold array, in pixels.</td>
</tr>
<tr>
<td>Thresholds</td>
<td>file</td>
<td><em>(Required)</em> When <code>sethalftone</code> is used to make a type 6 halftone dictionary the current dictionary, the next <code>width x height</code> characters are read from the file referenced by <code>file</code> and become the current threshold array. So <code>file</code> must reference a file open for read or read/write access at the time <code>sethalftone</code> is called. The <code>file</code> object can, of course, be the one returned by the <code>currentfile</code> operator. In that case, the next <code>width x height</code> characters are read from the input stream and saved as a threshold array. Also, <code>sethalftone</code> closes <code>file</code> if it encounters EOF and leaves it open otherwise.</td>
</tr>
<tr>
<td>TransferFunction</td>
<td>procedure</td>
<td>If present, overrides the transfer function specified by <code>settransfer</code> or <code>setcolortransfer</code>. Required in a type 6 halftone dictionary that is used as an element of a type 5 halftone dictionary for a non-primary color component.</td>
</tr>
<tr>
<td>Width</td>
<td>integer</td>
<td><em>(Required)</em> Width of the threshold array, in pixels.</td>
</tr>
</tbody>
</table>
The PostScript language has undergone several significant extensions. It is designed to be a universal standard for device-independent page descriptions, but each PostScript language implementation supports features and capabilities particular to that implementation. Appendix D, “Compatibility Strategies,” in the *PostScript Language Reference Manual, Second Edition*, presents guidelines for taking advantage of language extensions while maintaining compatibility with all PostScript interpreters.

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

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

Almost all of these functions are implemented as PostScript language procedures which call appropriate Level 2 operators such as `setpagedevice`.

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

The following is a list of the compatibility operators described in this chapter. They appear below in three groups by dictionary.

**Note** *In the following list, as well as the tables in this chapter, these symbols are used:*

- † = this compatibility operator is typically present in all 2012 PostScript implementations.
- ‡ = this compatibility operator is typically present in all 2012 PostScript implementations. However, in the absence of the associated feature, it performs no function aside from its documented effect on the operand stack.
- § = this compatibility operator requires execution in a system administrator job.
- ¶ = this compatibility operator can affect page device parameters.

Operators without a symbol are associated with a particular feature and are defined only if the feature is present in the product.

### In statusdict:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a3tray</td>
<td>†</td>
<td>appletalktype</td>
</tr>
<tr>
<td>b5tray</td>
<td>¶</td>
<td>byteorder†</td>
</tr>
<tr>
<td>checkpassword</td>
<td>†</td>
<td>diskonline</td>
</tr>
<tr>
<td>diskstatus</td>
<td>†</td>
<td>firstside</td>
</tr>
<tr>
<td>duplexmode</td>
<td>‡</td>
<td>pagecount‡</td>
</tr>
<tr>
<td>hardwareiomode</td>
<td>‡</td>
<td>processcolors</td>
</tr>
<tr>
<td>jobtimeout</td>
<td>†</td>
<td>realformat†</td>
</tr>
<tr>
<td>lettertray</td>
<td>¶</td>
<td>sccinteractive‡</td>
</tr>
<tr>
<td>margins</td>
<td>§</td>
<td>setsoftwareiomode‡</td>
</tr>
<tr>
<td>pagestackorder</td>
<td>§</td>
<td>settumble‡</td>
</tr>
<tr>
<td>product</td>
<td>†</td>
<td>tumble‡</td>
</tr>
<tr>
<td>revision</td>
<td>†</td>
<td>11x17tray †</td>
</tr>
<tr>
<td>setdefaulttimeouts</td>
<td>†§</td>
<td>a4tray‡</td>
</tr>
<tr>
<td>setduplexmode</td>
<td>¶</td>
<td>buildtime†</td>
</tr>
<tr>
<td>setmargins</td>
<td>¶</td>
<td>defaulttimeouts‡</td>
</tr>
<tr>
<td>setsccbatch</td>
<td>§</td>
<td>dostartpage</td>
</tr>
<tr>
<td>settumble</td>
<td>¶</td>
<td>dostartpage</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>dupletype</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>hardwareiomode‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>legaltray‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>manualfeed</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>manualfeedtimeout†</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>pagecount‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>processcolors</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>ramsize‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>sccbatch‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>setdefaulttimeouts§‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>setjobtimeout‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>setjobtimeout‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>setjobtimeout‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>setjobtimeout‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>setjobtimeout‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>setjobtimeout‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>setjobtimeout‡</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>¶</td>
<td>setjobtimeout‡</td>
</tr>
</tbody>
</table>

### In userdict:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a3</td>
<td>†</td>
<td>a4‡</td>
</tr>
<tr>
<td>b5</td>
<td>†</td>
<td>a4small‡</td>
</tr>
<tr>
<td>letter</td>
<td>¶</td>
<td>ledger‡</td>
</tr>
<tr>
<td>11x17</td>
<td>†</td>
<td>legal‡</td>
</tr>
</tbody>
</table>
In `systemdict`:

```
devdismount†§  devforall†  devformat§
devmount†§  devstatus†
```

### 6.2 Compatibility Operator Descriptions

This section describes the Level 1 compatibility objects present in Level 2 PostScript interpreters. The majority of these Level 1 objects are operators in `statusdict`. Other dictionaries may also contain compatibility objects (for example, `letter` in `userdict`). Compatibility objects need not always be operators (for example, the `waittimeout` integer in `statusdict`).

There is a Level 2 method of performing most Level 1 compatibility operations. For the following compatibility operators, there is currently no PostScript Level 2 equivalent:

```
checkpassword†  devforall†  emulate
firstside  newsheet  processcolors
sccinteractive‡  setpapertray§  setsccinteractive‡§
setuserdiskpercent§  userdiskpercent
```

The remaining compatibility objects are described below in terms of Level 2 operations. This not only provides the most accurate description of the compatibility operation but also indicates the correct Level 2 method of carrying out the operation.

Because many of the compatibility operations originally dealt with product-specific behavior, the semantics of some operations in Level 1 varied from one product to another. Defining compatibility operations in terms of product-independent Level 2 operations, corrects this problem at the cost of sometimes providing an imperfect emulation of the Level 1 operation.

Some Level 1 operations are no longer relevant for PostScript Level 2 language programs. In these cases, the compatibility operations may be implemented as no-ops that allow the PostScript Level 1 language program containing them to continue without generating errors. An example of such an operator is `setsccinteractive`.

#### 6.2.1 Error Behavior

In general, the behavior for error conditions is different between the Level 1 compatibility operation and the corresponding Level 2 method. This is to provide error behavior that is as similar to Level 1 error behavior as possible. As an example, a Level 1 paper tray operation
such as lettertray may generate a rangecheck while the corresponding Level 2 operation will generate a configurationerror or perform other actions under the control of Policies.

### 6.2.2 Using a Password to Change Persistent Values

In Level 1, many of the operations that changed persistent values could only be executed from jobs which had “exited the server” (this action required a password). If such an operation were executed without exiting the server an invalidaccess error resulted.

In Level 2, the notion of exiting the server has been replaced by the concept of an unencapsulated job (see section 3.7.7 of the PostScript Language Reference Manual, Second Edition). An unencapsulated job is entered by executing the Level 2 operator, startjob, or the Level 1 operator, exitserver. These operators require a password to be presented. The password must be equal to the value of either the StartJobPassword or the SystemParamsPassword system parameter. If the password is equal to the value of StartJobPassword, an ordinary unencapsulated job is started (see section 3.7.7 of the PostScript Language Reference Manual, Second Edition). If the password is equal to the value of SystemParamsPassword, a system administrator job is started. (If the SystemParamsPassword is a zero-length string or has never been set, every unencapsulated job is a system administrator job.)

Many compatibility operators change system or device parameters. Such operators use the Level 2 setsystemparams or setdevparams operators to emulate the Level 1 functionality. Those operators ordinarily require a Password parameter to be presented on each execution. This requirement is relaxed during a system administrator job, but not during an ordinary unencapsulated job. Since the compatibility operators do not present a password, this means they can be successfully executed only during a system administrator job. Executing them during an ordinary unencapsulated job (or any encapsulated job) will cause an invalidaccess error.

Compatibility operators that affect page device parameters save their persistent values only if they are executed from an unencapsulated job. In encapsulated jobs the values set by these compatibility operators will obey the normal save-restore rules and are not saved to persistent storage.

**Note** The compatibility objects are present in Level 2 printers for compatibility purposes only and their use in PostScript Level 2 language programs is strongly discouraged.
Table 6.1  statusdict compatibility operators

<table>
<thead>
<tr>
<th>Operator/Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a3tray</td>
<td>See section 6.2.5, “Paper Tray Operations.”</td>
</tr>
<tr>
<td>a4tray</td>
<td>See section 6.2.5, “Paper Tray Operations.”</td>
</tr>
<tr>
<td>appletalktype</td>
<td>– appletalktype string</td>
</tr>
<tr>
<td></td>
<td>is a string with the same value as the LocalTalkType device parameter in the %LocalTalk% parameter set and the EtherTalkType parameter in the %EtherTalk% parameter set. Redefining appletalktype will cause the LocalTalkType parameter to change as well as the EtherTalkType parameter. Similarly, changes to the EtherTalkType or the LocalTalkType parameter will change the string returned by the appletalktype operator.</td>
</tr>
<tr>
<td></td>
<td>The compatibility operator appletalktype is present only if either the %LocalTalk% or %EtherTalk% device names are present.</td>
</tr>
<tr>
<td></td>
<td>Errors: stackoverflow</td>
</tr>
<tr>
<td>b5tray</td>
<td>See section 6.2.5, “Paper Tray Operations.”</td>
</tr>
<tr>
<td>buildtime</td>
<td>– buildtime int</td>
</tr>
<tr>
<td></td>
<td>is an integer with the same value as the system parameter BuildTime.</td>
</tr>
<tr>
<td></td>
<td>Errors: stackoverflow</td>
</tr>
<tr>
<td>byteorder</td>
<td>– byteorder bool</td>
</tr>
<tr>
<td></td>
<td>is a boolean with the same value as the system parameter ByteOrder.</td>
</tr>
<tr>
<td></td>
<td>Errors: stackoverflow</td>
</tr>
<tr>
<td>checkpassword</td>
<td>int checkpassword bool</td>
</tr>
<tr>
<td></td>
<td>string checkpassword bool</td>
</tr>
<tr>
<td></td>
<td>checks whether string or int (int is converted to a string) is a valid password for either SystemParamsPassword or StartJobPassword. If valid, true is returned, otherwise false is returned. If either password is not set then true will be returned. A returned value of true indicates that string or int is a valid argument to startjob and exitserver. There is no PostScript Level 2 equivalent for checkpassword.</td>
</tr>
</tbody>
</table>
Errors: stackunderflow, typecheck

**defaulttimeouts** ‡

- **defaulttimeouts** *job manualfeed wait*

returns the system parameters **JobTimeout** and **WaitTimeout** and the page device parameter **ManualFeedTimeout** for *job*, *wait*, and *manualfeed* respectively. **defaulttimeouts** always returns three values, even if the corresponding system parameters are not present.

Errors: stackoverflow

**diskonline**

- **diskonline** *bool*

returns *true* if and only if a writeable disk device is mounted. This is determined by searching all device parameter sets named %disk*%, where * represents zero or more additional characters in the name. If the **Writeable** parameter is *true* for any of the sets searched, *bool* is set to *true*, otherwise it is set to *false*. Note that a disk parameter set with **Writeable** *true* need not have an initialized file system.

Errors: stackoverflow

**diskstatus**

- **diskstatus** *free total*

returns the number of disk pages (a page is 1024 characters) free and the total number of pages available on all writeable disk devices. This is determined by searching all device parameter sets named %disk*% that have a **Writeable** parameter set to *true*. The * represents zero or more additional characters in the name. *free* is the sum of the **Free** parameters from all such parameter sets, and *total* is the sum of the **LogicalSize** parameters from all such parameter sets.

Errors: stackoverflow

**dostartpage**

- **dostartpage** *bool*

returns the value of the system parameter **DoStartPage**.

The system parameter **DoStartPage** must be present for the compatibility operator **dostartpage** to be present.

Errors: stackoverflow
dosysstart

\[ \text{dosysstart bool} \]

returns \textit{false} if and only if the value of the system parameter \texttt{StartupMode} is 0.

The system parameter \texttt{StartupMode} must be present for the compatibility operator \texttt{dosysstart} to be present.

\textbf{Errors:} \hspace{1em} stackoverflow

\textbf{duplexmode}

See section 6.2.6, “Page Duplex Compatibility Operators.”

\textbf{emulate}

\[ \text{input-stream emulation-name emulate} \quad \text{or} \]

\[ \text{input-stream params-dict emulation-name emulate} \]

Causes the PostScript language interpreter to yield control, and the emulator named by \texttt{emulation-name} to start processing. The \texttt{emulate} operator is present in \texttt{statusdict}, and only in products which have one or more emulators co-resident with the PostScript language interpreter. The exact semantics of the emulators are product-dependent, and may be different in different products even though the emulation name may be the same. The specifics of each product’s emulators (if any) are documented in the product \textit{Addendum}. In most co-resident emulations, the command sequence \texttt{ESC-DEL-0} can be used to make the emulator yield control back to the PostScript interpreter; however, the PostScript context will generally have been lost.

The allowed values of \texttt{emulation-name} may be found in the implicit resource category \texttt{Emulator}. An illegal \texttt{emulation-name} will cause a rangecheck error.

A \texttt{params-dict} argument is optional. If the named emulator does not need parameters, and a \texttt{params-dict} is provided, the dictionary will be ignored.

If the named emulator requires parameters, and no \texttt{params-dict} is provided, then product-dependent defaults will be used if possible. Currently, no emulators require parameters.

The \texttt{input-stream} is a file object which becomes the input source for the emulator. The \texttt{input-stream} specified must be appropriate to the product-dependent emulator, as defined in the product \textit{Addendum}. An illegal \texttt{input-stream} will cause an invalidaccess error.

\textbf{Errors:} \hspace{1em} invalidaccess, rangecheck, stackoverflow, stackunderflow,
See section 6.2.6, “Page Duplex Compatibility Operators.”

hardwareiomode ‡

`hardwareiomode int`

returns an `int` which indicates the current communication channel whose corresponding device parameter set `Enabled` boolean is `true`. It will always return the channel indicated by `CurInputDevice` if that channel is on and enabled and one of the ones listed below. Otherwise, the smallest such `int` is returned. If none in the list are on and enabled, 0 is returned. The interpretation of `int` is:

<table>
<thead>
<tr>
<th><code>int</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><code>%Serial%</code></td>
</tr>
<tr>
<td>1</td>
<td><code>%Parallel%</code></td>
</tr>
<tr>
<td>2</td>
<td><code>%LocalTalk%</code></td>
</tr>
<tr>
<td>3</td>
<td><code>%SerialB%</code></td>
</tr>
</tbody>
</table>

The `Serial`, `Parallel`, `SerialB`, or `LocalTalk` device parameter set must be present for the compatibility operator `hardwareiomode` to be present.

Errors: `stackoverflow`

initializedisk §

`pages action initializedisk` –

initializes each writeable disk, setting the disk device parameters `LogicalSize` and `InitializeAction` to the value of `pages` and `action+1`, respectively.

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

jobname †

`jobname string`

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

The user parameter `JobName` must be present for the compatibility operator `jobname` to be present.

Errors: `stackoverflow`
jobtimeout †

– jobtimeout int

returns the value of the user parameter JobTimeout.

Errors: stackoverflow

ledgertray¶

See section 6.2.5, “Paper Tray Operations.”

legaltray¶

See section 6.2.5, “Paper Tray Operations.”

lettertray †¶

See section 6.2.5, “Paper Tray Operations.”

manualfeed

– manualfeed bool

is a boolean that works in conjunction with the page device parameter ManualFeed to determine whether a page is fed manually. If either manualfeed or ManualFeed is true at the time of a showpage or copypage then that page will be fed manually; otherwise the page will not be fed manually.

The values of ManualFeed and manualfeed are determined independently. That is, setting the manualfeed boolean or setting the pagedevice parameter, ManualFeed, does not affect the value of the other.

The manualfeed key is present in statusdict if and only if the page device parameter ManualFeed is defined for the product. The initial value of manualfeed at power-on is false.

Errors: stackoverflow

manualfeedtimeout

– manualfeedtimeout int

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

The values of ManualFeedTimeout and manualfeedtimeout are determined independently. That is, setting the manualfeedtimeout integer or setting the pagedevice parameter, ManualFeedTimeout, does not effect the value of the other.

Errors: stackoverflow
margins ‡

Returns the x and y components of the page device parameter **Margins** as *left* and *top*, respectively.

**Errors:** stackoverflow

**newsheet**

See section 6.2.6, “Page Duplex Compatibility Operators.”

**pagecount ‡**

Returns the value of the system parameter **PageCount**.

**Errors:** stackoverflow

**pagestackorder**

Returns the logical complement of the page device **OutputFaceUp** boolean parameter. For example, if **OutputFaceUp** is *true*, **bool** will be *false*.

The page device parameter **OutputFaceUp** must be present for the compatibility operator **pagestackorder** to be present.

**Errors:** stackoverflow

**printername ‡**

Stores the value of the system parameter **PrinterName** in **string** and returns a string object designating the *substring* actually used.

**Errors:** rangecheck, stackunderflow, typecheck

**processcolors**

Returns the number of device process color components in the current page device (1 for black, 3 for RGB or CMY, or 4 for CMYK). This compatibility operator is only present for devices capable of color printing (which have more than one process color to print). Its absence indicates a monochrome-only device (1 process color).

**Errors:** stackoverflow,
product $^\dagger$ – product string

is a string in statusdict initialized to the value of the string product in systemdict.

Errors: stackoverflow

ramsize – ramsie int

returns the number of bytes of RAM available to the product. Refer to the RamSize system parameter.

Errors: stackoverflow

realformat $^\dagger$ – realformat string

is a string with the same value as the system parameter RealFormat.

Errors: stackoverflow

revision $^\dagger$ – revision int

is an integer with the same value as the system parameter Revision.

Errors: stackoverflow

sccbatch

See section 6.2.3, “SCC Operations.”

sccinteractive $^\ddagger$

See section 6.2.3, “SCC Operations.”

setdefaulttimeouts $^\dagger$§¶

job manualfeed wait setdefaulttimeouts –

This compatibility operator sets the system parameters JobTimeout and WaitTimeout to job and wait, respectively, and sets the page device parameter ManualFeedTimeout to manualfeed. setdefaulttimeouts always takes three values, even if the corresponding system or pagedevice parameters are not present.

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

*bool setdostartpage* –

This compatibility operator sets the system parameter *DoStartPage* to the value of *bool*.

The system parameter *DoStartPage* must be present for the compatibility operator *setdostartpage* to be present.

Errors: invalidaccess, stackunderflow, typecheck

**setdosysstart**

*bool setdosysstart* –

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

The system parameter *StartupMode* must be present for the compatibility operator *setdosysstart* to be present.

Errors: invalidaccess, stackunderflow, typecheck

**sethardwareiomode**

*int sethardwareiomode*

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

<table>
<thead>
<tr>
<th>int</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Open %Serial% and %SerialB%. Close all others.</td>
</tr>
<tr>
<td>1</td>
<td>Open %Parallel%. Close all others.</td>
</tr>
<tr>
<td>2</td>
<td>Open %LocalTalk% and %EtherTalk% (if both exist). Close all others.</td>
</tr>
<tr>
<td></td>
<td>Open %LocalTalk% (if only %LocalTalk% exists). Close all others.</td>
</tr>
<tr>
<td></td>
<td>Open %EtherTalk% (if only %EtherTalk% exists). Close all others.</td>
</tr>
<tr>
<td>3</td>
<td>Open %Serial% and %SerialB%. Close all others.</td>
</tr>
</tbody>
</table>

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

*int setjobtimeout* –

This compatibility operator sets the user parameter `JobTimeout` to the value of `int`.

The user parameter `JobTimeout` must be present for the compatibility operator `setjobtimeout` to be present.

**Errors:** stackunderflow, typecheck

---

**setmargins** ‡§†

*top left setmargins* –

This compatibility operator sets the page device `Margins` parameter to `[left top]`.

The page device parameter `Margins` must be present for the compatibility operator `setmargins` to be present.

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

---

**setpagestackorder** §§

*bool setpagestackorder* –

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

The page device parameter `OutputFaceUp` must be present for the compatibility operator `setpagestackorder` to be present.

**Errors:** invalidaccess, stackunderflow, typecheck

---

**setprintname** ‡§

*string setprintname* –

This compatibility operator sets the system parameter `PrinterName` to the value of `string`.

The system parameter `PrinterName` must be present for the compatibility operator `setprintname` to be present.

**Errors:** invalidaccess, limitcheck, stackunderflow, typecheck

---

**sccbatch**

See section 6.2.3, “SCC Operations.”
**sccinteractive**‡  
See section 6.2.3, “SCC Operations.”

**setsoftwareiomode**‡§  

```
int setsoftwareiomode –
```

This compatibility operator sets the values of the **Interpreter**, and if appropriate, **Protocol** device parameters for the current communications device parameter set (as indicated by the system parameter **CurInputDevice**). The meaning of *int* is:

<table>
<thead>
<tr>
<th>int</th>
<th>Interpreter value</th>
<th>Protocol value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><strong>Interpreter</strong>: PostScript;</td>
<td><strong>Protocol</strong>: Normal</td>
</tr>
<tr>
<td>1</td>
<td><strong>Interpreter</strong>: ProprinterXL;</td>
<td><strong>Protocol</strong>: Raw</td>
</tr>
<tr>
<td>2</td>
<td><strong>Interpreter</strong>: Diablo630;</td>
<td><strong>Protocol</strong>: Raw</td>
</tr>
<tr>
<td>3</td>
<td><em>Reserved.</em></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><strong>Interpreter</strong>: HP7475A;</td>
<td><strong>Protocol</strong>: Raw</td>
</tr>
<tr>
<td>5</td>
<td><strong>Interpreter</strong>: LaserJetIIIP;</td>
<td><strong>Protocol</strong>: Raw</td>
</tr>
<tr>
<td></td>
<td>if the LaserJetIIIP emulator is present in the product.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>Interpreter</strong>: LaserJetIII;</td>
<td><strong>Protocol</strong>: Raw</td>
</tr>
<tr>
<td></td>
<td>if the LaserJetIII emulator is present in the product.</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td><strong>Interpreter</strong>: PostScript;</td>
<td><strong>Protocol</strong>: Binary</td>
</tr>
</tbody>
</table>

*Note: A product will probably never have both the LaserJetIIIP and LaserJetIII emulators installed. If a product does have both emulators installed, passing a value of 5 to *setsoftwareiomode* will select only LaserJetIIIP.*

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

**settumble**‡  
See section 6.2.6, “Page Duplex Compatibility Operators.”

**setuserdiskpercent**‡  

```
int setuserdiskpercent –
```

pops *int* off the stack. This operator is essentially a no-op.

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

**softwareiomode**‡  

```
– softwareiomode int
```

returns *int* which indicates (see *setsoftwareiomode*) the interpretation mode for the current communications device (as indicated by the system parameter **CurInputDevice**).
Errors: stackoverflow

**tumble**

See section 6.2.6, “Page Duplex Compatibility Operators.”

**usertext**

– **usertext** *int*

returns the value 0. This operator is essentially a no-op.

Errors: stackoverflow

**waittimeout †**

– **waittimeout** *int*

is an integer with the same value as the user parameter **WaitTimeout**. Redefining either **waittimeout** or the user parameter **WaitTimeout** redefines the other to the same value.

The user parameter **WaitTimeout** must be present for the compatibility operator **waittimeout** to be present.

Errors: stackoverflow

**11x17tray‡**

See section 6.2.5, “Paper Tray Operations.”

---

### 6.2.3 SCC Operations

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

<table>
<thead>
<tr>
<th>position 7</th>
<th>stop bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 stop bit</td>
</tr>
<tr>
<td>1</td>
<td>2 stop bits</td>
</tr>
</tbody>
</table>

---
Table 6.3  *Data bits*

<table>
<thead>
<tr>
<th>positions 6 and 5</th>
<th>data bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>standard</td>
</tr>
<tr>
<td>1</td>
<td>7 bits</td>
</tr>
<tr>
<td>2</td>
<td>8 bits</td>
</tr>
</tbody>
</table>

Table 6.4  *Flow control*

<table>
<thead>
<tr>
<th>positions 4, 3 and 2</th>
<th>Flow control</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Xon/Xoff</td>
</tr>
<tr>
<td>1</td>
<td>Dtr</td>
</tr>
<tr>
<td>2</td>
<td>Etx/Ack</td>
</tr>
</tbody>
</table>

Table 6.5  *Parity*

<table>
<thead>
<tr>
<th>positions 1 and 0</th>
<th>parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Space</td>
</tr>
<tr>
<td>1</td>
<td>Odd</td>
</tr>
<tr>
<td>2</td>
<td>Even</td>
</tr>
<tr>
<td>3</td>
<td>Mark</td>
</tr>
</tbody>
</table>

In Level 1, the data bits and parity interacted in a non-orthogonal manner to produce a table of possible choices for data and parity that included many common desired methods of sending data. The “standard” data bits setting was only present for backward compatibility purposes with earlier versions of the SCC operators. In particular, a standard data bit setting could always be achieved with either a 7- or 8-bit data setting. In Level 2, there are analogous entries as above for the %Serial% and %SerialB% device parameter sets.

The mapping between Level 1 stop bits and flow control and Level 2 device parameters **StopBits** and **FlowControl**, respectively, is straightforward and obvious. It is not possible to provide such a one to one correspondence between the Level 1 notion of data bits and parity and the Level 2 device parameters **DataBits** and **Parity**. Tables 6.6 and 6.7 show the conversion between Level 1 data bits and parity and Level 2 **DataBits** and **Parity**. Notice that in going from **DataBits** and **Parity** to data bits and parity, standard parity is never used.
Table 6.6  Options byte to devparams conversion

data bits & parity --> DataBits & Parity

| standard space | 7 bits Space |
| standard mark  | 8 bits None  |
| standard odd   | 7 bits Odd   |
| standard even  | 7 bits Even  |
| 7 bits space   | 7 bits Space |
| 7 bits mark    | 7 bits Mark  |
| 7 bits odd     | 7 bits Odd   |
| 7 bits even    | 7 bits Even  |
| 8 bits space   | 8 bits None  |
| 8 bits mark    | 8 bits None  |
| 8 bits odd     | 8 bits Odd   |
| 8 bits even    | 8 bits Even  |

Table 6.7  devparams to options byte conversion

DataBits & Parity --> data bits & parity

| 7 bits None   | 7 bits mark |
| 7 bits Space  | 7 bits space|
| 7 bits Mark   | 7 bits mark |
| 7 bits Odd    | 7 bits odd  |
| 7 bits Even   | 7 bits even |
| 8 bits None   | 8 bits mark |
| 8 bits Space  | 8 bits space|
| 8 bits Mark   | 8 bits mark |
| 8 bits Odd    | 8 bits odd  |
| 8 bits Even   | 8 bits even |

These tables are defined to provide the best compatibility with Level 1 behavior. In several cases, no correct choice is possible. For example, in Level 1 there was no support for 7 data bits with no parity (that is, the total number of data and parity bits is 7). The Level 2 setting of 7 bits None is imperfectly mapped to 7 bits mark. Most serial hardware does not support 8-bit Mark or Space and for this reason these values are never generated in mapping from Level 1 to Level 2. In fact, in Level 1, 8 bits mark and space actually provided the equivalent of the Level 2 8 bits None functionality.
Table 6.8  SCC compatibility operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>sccbatch</td>
<td>channel sccbatch baud options</td>
</tr>
<tr>
<td></td>
<td>returns the serial communications device parameter settings. The values are from either the %SerialB_NV% (if channel equals 9) or the %Serial_NV% (if channel equals 25) parameter set. The value of options is encoded as described above and the values for data bits and parity are determined by Table 6.7. The values for baud, stop bits, and flow control are determined from the corresponding settings for the Baud, StopBits, and FlowControl device parameters, respectively.</td>
</tr>
</tbody>
</table>

The %Serial_NV% or %SerialB_NV%, device parameter set must be present for the compatibility operator sccbatch to be present.

Errors: rangecheck, stackoverflow, stackunderflow, typecheck

sccinteractive ‡  channel sccinteractive baud options
|              | pops the input argument off the stack and pushes 0 0 on the stack. This operator is essentially a no-op. |
|              | Errors: invalidaccess, rangecheck, stackoverflow, stackunderflow, typecheck |

setsccbatch §  channel baud options setsccbatch –
|              | This compatibility operator sets the communication device parameters for serial communications. Either the %SerialB_NV% (if channel equals 9) or the %Serial_NV% (if channel equals 25) settings are affected. The following device parameters are affected by baud and options: Baud, StopBits, DataBits, FlowControl, Parity, and CheckParity. Baud, StopBits, and FlowControl are set according to the corresponding values for baud, stop bits and flow control. DataBits and Parity are set based on Table 6.7 above. CheckParity is set according to the new Parity setting: |
|              | • true if the setting is Odd or Even. |
|              | • false if the setting is Space or Mark. |
|              | • not changed if the setting is None (parity checking is not done if Parity is None independent of the setting of CheckParity). |

The %Serial_NV% or %SerialB_NV%, device parameter set must be present for the compatibility operator setsccbatch to be present.
Errors: invalidaccess, rangecheck, stackunderflow, typecheck

`setsccinteractivestates` channel baud options `setsccinteractivestates` –

pops the three input arguments off the stack. This operator is essentially a no-op.

Errors: rangecheck, stackunderflow, typecheck

6.2.4 Paper Size Operations

All the operators in this section are in `userdict`. Each operator executes `setpagedevice` to request a specific paper size. The only difference among these operations is the size of paper requested and the Imaging-BBox. The “–small” operators specify a non-null ImagingBBox while the non-small operators specify a null ImagingBBox. These operators use the specified size as indicated below as a page device PageSize parameter. In addition, all these operators set the PageSize Policy to 7, which guarantees that the imaging area established is the requested size regardless of the medium’s actual size and turns off the normal PostScript Level 2 media matching mechanism. The only error that is generated is a limitcheck caused by insufficient memory for the requested imaging area. In Table 6.9, default units (1/72 inch) are used as the units for the PageSize and ImagingBBox.

Table 6.9 Paper size compatibility operators (in userdict)

<table>
<thead>
<tr>
<th>Operator</th>
<th>PageSize</th>
<th>ImagingBBox</th>
</tr>
</thead>
<tbody>
<tr>
<td>letter ‡¶</td>
<td>[612 792]</td>
<td>null</td>
</tr>
<tr>
<td>letter small ‡¶</td>
<td>[612 792]</td>
<td>[25 25 587 767]</td>
</tr>
<tr>
<td>legal ‡¶</td>
<td>[612 1008]</td>
<td>null</td>
</tr>
<tr>
<td>ledger ‡¶</td>
<td>[1224 792]</td>
<td>null</td>
</tr>
<tr>
<td>11x17 ‡¶</td>
<td>[792 1224]</td>
<td>null</td>
</tr>
<tr>
<td>a4 ‡¶</td>
<td>[595 842]</td>
<td>null</td>
</tr>
<tr>
<td>a3 ‡¶</td>
<td>[842 1191]</td>
<td>null</td>
</tr>
<tr>
<td>a4small ‡¶</td>
<td>[595 842]</td>
<td>[25 25 570 817]</td>
</tr>
<tr>
<td>b5 ‡¶</td>
<td>[516 729] or [499 709]</td>
<td>null</td>
</tr>
<tr>
<td>note ‡¶</td>
<td>[width height]</td>
<td>[25 25 width-25]</td>
</tr>
</tbody>
</table>
The **note** compatibility operator will be present only if the size \([width \ height]\) is an element of the **PageSize** array in some instance of the **OutputDevice** resource category.

The **letter** and **lettersmall** compatibility operators will be present only if the size \([612 \ 792]\) is an element of the **PageSize** array in some instance of the **OutputDevice** resource category.

The **legal** compatibility operator will be present only if the size \([612 \ 1008]\) is an element of the **PageSize** array in some instance of the **OutputDevice** resource category.

The **a4** and **a4small** compatibility operators will be present only if the size \([595 \ 842]\) is an element of the **PageSize** array in some instance of the **OutputDevice** resource category.

The **b5** compatibility operator will be present only if the size \([516 \ 729]\) or the size \([499 \ 709]\) is an element of the **PageSize** array in some instance of the **OutputDevice** resource category.

### 6.2.5 Paper Tray Operations

All of the operators in this section are in **statusdict**. Each operator executes **setpagedevice** to request a tray containing a specific paper size. The only difference among these operations is the size of paper requested. The **PageSize** requested is the same as for the corresponding page size operator discussed in the previous section and the **ImagingBBox** requested is always null. These operators use the specified size as indicated below as a page device **PageSize** parameter.

All of these operators set the **PageSize Policy** to 0, which guarantees that a **configurationerror** is generated if a tray containing the requested paper size is not present. The implementation of the compatibility operators convert any such **configurationerror** to a **rangecheck** for compatibility with PostScript Level 1 implementations. Also, a **limitcheck** error can occur because of insufficient memory for the requested imaging area.

### Table 6.10 Paper tray compatibility operators

<table>
<thead>
<tr>
<th>Operator</th>
<th><strong>PageSize</strong></th>
<th><strong>ImagingBBox</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>lettertray</td>
<td>[612 792]</td>
<td>null</td>
</tr>
<tr>
<td>legaltray</td>
<td>[612 1008]</td>
<td>null</td>
</tr>
<tr>
<td>ledgertray</td>
<td>[1224 792]</td>
<td>null</td>
</tr>
<tr>
<td>a3tray</td>
<td>[842 792]</td>
<td>null</td>
</tr>
<tr>
<td>a4tray</td>
<td>[595 842]</td>
<td>null</td>
</tr>
</tbody>
</table>
The `lettertray` compatibility operator will be present only if the size [612 792] is an element of the `PageSize` array in some instance of the `OutputDevice` resource category.

The `legaltray` compatibility operator will be present only if the size [612 1008] is an element of the `PageSize` array in some instance of the `OutputDevice` resource category.

The `a4tray` compatibility operator will be present only if the size [595 842] is an element of the `PageSize` array in some instance of the `OutputDevice` resource category.

The `b5tray` compatibility operator will be present only if the size [516 729] or the size [499 709] is an element of the `PageSize` array in some instance of the `OutputDevice` resource category.

### 6.2.6 Page Duplex Compatibility Operators

All compatibility objects described below are defined in `statusdict` unless otherwise specified.

#### Table 6.11 Page duplex compatibility operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>duplexmode</td>
<td>– <code>duplexmode bool</code></td>
</tr>
</tbody>
</table>

returns the value of the page device parameter `Duplex`.

The page device parameter `Duplex` must be present for the compatibility operator `duplexmode` to be present.

**Errors:** `stackoverflow`

| firstside | – `firstside bool`  |

returns `true` if the current page is a front side, `false` if the current page is a back side.

The page device parameter `Duplex` must be present for the compatibility operator `firstside` to be present.

**Errors:** `stackoverflow`
newsheet

If Duplex is true and the current page is a back-side, causes this page to be printed as is (perhaps blank) and sets up a clean printing environment for the next page. Otherwise, executing newsheet has no affect.

The page device parameter Duplex must be present for the compatibility operator newsheet to be present.

Errors: none

setduplexmode

bool setduplexmode –

This compatibility operator sets the page device parameter Duplex to bool.

The page device parameter Duplex must be present for the compatibility operator setduplexmode to be present.

Errors: stackunderflow, typecheck

settumble

bool settumble –

This compatibility operator sets the page device parameter Tumble to bool.

The page device parameter Duplex must be present for the compatibility operator settumble to be present.

Errors: stackunderflow, typecheck

tumble

– tumble bool

returns the value of the page device parameter Tumble.

The page device parameter Duplex must be present for the compatibility operator tumble to be present.

Errors: stackoverflow

6.2.7 Device Compatibility Operators

All device compatibility operators described below are defined in systemdict. The device operators aid in management of any given file system.
Table 6.12  Device compatibility operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>devdismount †‡</td>
<td>string devdismount –</td>
</tr>
<tr>
<td></td>
<td>This compatibility operator sets to false the Mounted device parameter</td>
</tr>
<tr>
<td></td>
<td>boolean of the parameter set corresponding to the device specified by</td>
</tr>
<tr>
<td></td>
<td>string. It is necessary for the device to be mounted before it can be</td>
</tr>
<tr>
<td></td>
<td>dismounted. Trying to dismount a device that is not mounted will have no</td>
</tr>
<tr>
<td></td>
<td>affect. Some devices cannot be dismounted. Trying to dismount these</td>
</tr>
<tr>
<td></td>
<td>will also have no affect.</td>
</tr>
<tr>
<td></td>
<td>In PostScript Level 2, you can dismount from any save level if the</td>
</tr>
<tr>
<td></td>
<td>SystemParamsPassword is not set. If it is set, devdismount will raise an</td>
</tr>
<tr>
<td></td>
<td>invalidaccess error unless executed within a unencapsulated system</td>
</tr>
<tr>
<td></td>
<td>administrator job.</td>
</tr>
<tr>
<td>devdismount</td>
<td>raises a configurationerror if any other type of error occurs during a</td>
</tr>
<tr>
<td></td>
<td>dismount.</td>
</tr>
<tr>
<td></td>
<td>Errors: configurationerror, invalidaccess,</td>
</tr>
<tr>
<td></td>
<td>stackunderflow, undefinedfilename</td>
</tr>
<tr>
<td>devforall †</td>
<td>proc scratch devforall –</td>
</tr>
<tr>
<td>devforall</td>
<td>enumerates all known storage devices.</td>
</tr>
<tr>
<td></td>
<td>For each storage device, devforall copies its name into the supplied</td>
</tr>
<tr>
<td></td>
<td>scratch string, pushes a string object that is the substring of scratch</td>
</tr>
<tr>
<td></td>
<td>that was actually used, and calls proc. devforall does not return any</td>
</tr>
<tr>
<td></td>
<td>results of its own, but proc may do so.</td>
</tr>
<tr>
<td></td>
<td>Errors: invalidaccess, rangecheck, stackoverflow,</td>
</tr>
<tr>
<td></td>
<td>stackunderflow, typecheck, undefined</td>
</tr>
<tr>
<td>devformat †‡</td>
<td>string pages format devformat -</td>
</tr>
<tr>
<td>devformat</td>
<td>This compatibility operator sets the LogicalSize device parameter of</td>
</tr>
<tr>
<td></td>
<td>the parameter set corresponding to the device specified by string to the</td>
</tr>
<tr>
<td></td>
<td>value specified by pages. It then sets the InitializeAction, in the same</td>
</tr>
<tr>
<td></td>
<td>parameter set, to the value of format+1. Refer to the InitializeAction</td>
</tr>
<tr>
<td></td>
<td>and LogicalSize file system device parameters for complete details.</td>
</tr>
<tr>
<td></td>
<td>Errors: invalidaccess, limitcheck, rangecheck,</td>
</tr>
<tr>
<td></td>
<td>stackunderflow, typecheck, undefined,</td>
</tr>
<tr>
<td></td>
<td>undefinedfilename,</td>
</tr>
</tbody>
</table>

6.2 Compatibility Operator Descriptions
**devmount**

This compatibility operator sets to *true* the `Mounted` device parameter boolean of the parameter set corresponding to the device specified by `string`. It then returns the resulting value of `Mounted` by reading it from the same parameter set. *True* indicates that the device was successfully mounted or was already mounted. *False* indicates that the device cannot be mounted at this time.

In PostScript Level 2, you can mount from any save level if the `SystemParamsPassword` is not set. If it is set, `devmount` will raise an `invalidaccess` error unless executed within a unencapsulated system administrator job.

**Errors:** `invalidaccess`, `stackunderflow`, `undefinedfilename`

**devstatus**

Takes a device name identified by `string` from the stack. If the device name is unknown, *false* will be left on the stack only. If the device name is found, it pushes various file system attributes for the device. The attributes are `searchable`, `writeable`, `hasNames`, `mounted`, `removable`, `searchOrder`, `freePages`, and `size`.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>searchable</code></td>
<td>The <code>searchable</code> attribute corresponds to the <code>Searchable</code> device parameter and is a boolean which indicates that the device will be searched when looking for a file with no device name prefix in its name.</td>
</tr>
<tr>
<td><code>writeable</code></td>
<td>The <code>writeable</code> attribute corresponds to the <code>Writeable</code> device parameter and indicates whether files on this device can be written.</td>
</tr>
<tr>
<td><code>hasNames</code></td>
<td>The <code>hasNames</code> attribute corresponds to the <code>HasNames</code> device parameter and is a boolean which indicates whether the device supports named files.</td>
</tr>
<tr>
<td><code>mounted</code></td>
<td>The <code>mounted</code> boolean (<code>Mounted</code> device parameter) indicates whether the device is mounted.</td>
</tr>
<tr>
<td><code>removable</code></td>
<td>The <code>removable</code> boolean (<code>Removable</code> device parameter) indicates whether the media within the device can be removed.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>searchOrder</code></td>
<td>The <code>searchOrder</code> attribute (<code>SearchOrder</code> device parameter) indicates the priority at which the device participates when searching for a file in operations in which no device has been specified.</td>
</tr>
<tr>
<td><code>freePages</code></td>
<td>The <code>freePages</code> boolean (<code>Free</code> device parameter) indicates the amount of free space (in pages).</td>
</tr>
<tr>
<td><code>size</code></td>
<td>The <code>size</code> attribute (<code>Size</code> device parameter) indicates the current size of the PostScript file system (in pages).</td>
</tr>
</tbody>
</table>

*Note: In Level 1, a “page” had a file system specific size (typically 1024).*

For a complete description of each of the device parameters mentioned above, refer to section 3.5.3.

**Errors:** stackunderflow
A.1 Changes since Version 2011, January 24, 1992

The following is a brief list of changes made to the PostScript Language Reference Manual Supplement For Version 2011 since January 24, 1992. These changes were incorporated into the 2012 release.

- A new chapter has been added, titled “Other Extensions to PostScript Level 2.” This chapter has a description of the new Type 6 Halftone Dictionary.

- Corrections have been made throughout the document. In addition, the text has been expanded in many places to provide more context for the reader.

- The new Page Device Parameters introduced in 2012 are DeviceRenderingInfo, FaxOptions, ProcessColorModel, SeparationColorNames, and SeparationOrder.

- The Fax feature is described in detail in sections 2.2 and 3.6.

- Section 3.1 and 3.2 now give the reader better context when thinking about unencapsulated jobs and passwords.

- There is a new User Parameter introduced in 2012 called Accurate-Screens.

- There are several new System Parameters introduced in 2012. They are CurStoredFontCache, CurStoredScreenCache, MaxHWRendering-Buffer, MaxImageBuffer, MaxStoredFontCache, and MaxStored-ScreenCache.

- There are several new Device Parameter sets. The list includes %Scsi-Comm%, %os%, %Scsi%, %Engine%, %LaserJetIII%, %Fax%, and %Calendar%. 
• The **Interpreter** key in all of the Device Parameter sets of type `/Communications` can now be set to **LaserJetIII** for PCL5 emulation. **EpsonFX850** is also a new choice.

• The **Interpreter** key in all of the Device Parameter sets of type `/Communications` can be set to **AutoSelect**. When set to **AutoSelect**, automatic and seamless switching between the available interpreters and emulators is enabled.

• In the serial device parameter set, a new **FlowControl** choice is available called **XonXoff2**.

• A new **Protocol** choice has been added for serial and parallel parameter sets called **TBCP**.

• The new key **Filtering** has been added to the LocalTalk set.

• The new keys **Handshake** and **OutputDevice** have been added to the parallel set.

• The device parameter sets of type `/FileSystem` have been changed to support removable media. A new key named **BlockSize** has been added. The `%disk%` parameter sets have the new keys **Bus** and **Interleave** added.

• The new implicit resource category **HWOptions** has been added. There are some implicit **ProcSets** for fax called **FaxOps** and **FaxAdminOps**. Also, **HalftoneType** can be set to 6.

• In the compatibility chapter, **processcolors** has been added.
Index

A
a3 98, 115
a3tray 98, 101, 116
a4 98, 115, 116
a4small 98, 115, 116
a4tray 98, 101, 116, 117
a5tray 98, 101, 117
AccurateScreens 37
ActivityReport 81
AdvanceDistance 5
AdvanceMedia 5
appleTalktype 60, 62, 98, 101
AutoLF 80

B
b5 115, 116
Baud 55, 114
BeginPage 5
Bind 6
BindDetails 6
BlockSize 65, 69
BoldFontName 80
Booklet 6
BookletDetails 6
BootDelay 73
BSizeStandard 75
BuildTime 39, 101
buildtime 98, 101
Bus 65
ByteOrder 39, 101
byteorder 98, 101

C
calendar device parameters
  Day 86
  Hour 86
  Minute 86
  Month 86
  Running 86
  Second 86
  Year 87
CallCount 88
CalleeID 25
CalleePhone 19, 23, 24
CallerID 19
CallerPhone 19
CallLength 25, 89
Calls 88
Calls dictionary entries
  CallLength 89
  CoverPagesSent 89
  ErrorArray 89
  ErrorIndex 89
  FaxKind 89
  Pages 90
  PagesSent 90
  TimeSent 90
CartridgeID 70
CartridgeType 70
Category 42, 93
CheckParity 56, 73, 114
checkpassword 98, 99, 101
Collate 5
ColorRendering 42, 91
ColorRenderingType 92
ColorSetup 79
ColorSpace 42, 91
ColorSpaceFamily 92
compatibility operators 2
  11x17 98, 115
  11x17tray 98, 111, 117
  a3 98, 115
  a3tray 98, 101, 116
  a4 98, 115, 116
a4small 98, 115, 116
a4tray 98, 101, 116, 117
appletalktype 60, 62, 98, 101
b5 98, 115, 116
b5tray 98, 101, 117
buildtime 98, 101
byteorder 98, 101
checkpassword 98, 99, 101
defaulttimeouts 98, 102
devdismount 99, 119
devforall 99, 119
devformat 99, 119
devmount 99, 120
devstatus 99, 120
diskonline 98, 102
diskstatus 98, 102
dostartpage 98, 102
dosysstart 98, 102
duplexmode 98, 103, 117
emulate 98, 99, 103
exitserver 35, 46, 100, 101
firstside 98, 99, 104, 117
hardwareiomode 98, 104
initializedisk 98, 104
jobname 98, 104
jobtimeout 98, 105
ledger 98, 115
ledgertray 98, 105, 116
legal 98, 115, 116
legaltray 98, 105, 116
letter 98, 115, 116
lettersmall 98, 115, 116
lettertray 98, 105, 116, 117
manualfeed 98, 105
manualfeedtimeout 98, 105
margins 98, 106
newsheet 98, 99, 106, 118
note 98, 115, 116
pagecount 98, 106
pagestackorder 98, 106
printername 98, 106
processcolors 98, 99, 106
product 98, 107
ramsize 98
revision 98, 107
scchatch 98, 109, 114
scинтерактив 98, 99, 107, 110
setdefaulttimeout 98, 107
setdostartpage 98
setdostysstart 98, 108
setduplexmode 98, 118
sethardwareiomode 98, 108
setjobtimeout 98, 109
setmargins 98, 109
setpagestackorder 98, 109
setpapertextray 99
setprintername 98, 109
setscchatch 98, 114
scинтерактив 98, 99, 115
setsoftwareiomode 98, 110
settumble 98, 110, 118
setuserdiskpercent 98, 99, 110
startjob 35, 46, 100, 101
tumble 98, 111, 118
userdiskpercent 98, 99, 111
waittimeout 98, 111
configurationerror 17, 31, 48, 55, 58, 68, 74, 100, 116
Confirmation 19, 25
Copies 20, 27, 28, 78
CoverNote 20
CoverPagesSent 89
CoverSheet 19, 20, 24, 25, 26, 27
CoverSheetOnly 21, 28
CoverType 25
CurDisplayList 39
CurFontCache 39
CurFormCache 39
CurInputDevice 39, 110
CurOutlineCache 39
CurOutputDevice 39
CurPatternCache 39
currentdevparams operator 2, 35, 47, 65, 77, 80, 87
currenthalftone operator 95
currentpagedevice operator 2, 5
currentPageNo 25
currentsystemparams operator 2, 35, 38
currentuserparams operator 2, 35
curScreenStorage 39
curSourceList 40
curUPathCache 39
cutMedia 5

D

Darkness 75
DataBits 56, 112, 114
Index

FaxSendPS operator 17, 18, 23, 24, 27, 28, 83, 84
FaxType 21, 27, 81
Filter 92
Filtering 60
findresource operator 87
firstside 98, 99, 104, 117
FlowControl 56, 112, 114
FMapType 92
Fold 6
FoldDetails 7
Font 41, 91
FontFixed 78
FontHeight 78
FontItalic 78
FontPitch 78
FontResourceDir 41
FontSymbolSet 79
FontType 92
FontTypeface 79
FontWeight 79
Form 42
FormType 92
Free 66, 70, 102, 121

G
Generic 93
GenericResourceDir 41, 42, 43
GenericResourcePathSep 42

H
Halftone 42, 92
halftone dictionary
  type 6 95
HalftoneType 92, 96
Handshake 63
hardwareiomode 98, 104
HasNames 53, 57, 60, 62, 63, 64, 66, 70, 72, 120
Height 96
Hour 86
HWOptions 92
HWResolution 5, 8, 94
  instance of OutputDevice resource category 94
I
ID 19, 24, 82
ImageType 92
ImagingBBox 5, 115, 116
IncludesFinalPage 26
InitializeAction 66, 67, 70, 72, 104, 119
initializedisk 98, 104
InitialPage 26
InitiatorID 74
InputAttributes 7, 16
InsertSheet 7
Install 5
Interleave 66, 67
Interpreter 54, 55, 57, 58, 60, 62, 63, 64, 76, 110
invalidaccess 33, 100, 119, 120
ioerror 56

J
job dictionary entries
  CallCount 88
  Calls 88
  DialCallee 88
  ErrorArray 88
  ErrorIndex 88
  JobId 88
  RecipientName 88
  RecipientOrg 88
  RecipientPhone 88
  SenderName 89
  SenderOrg 89
  TimeBegan 89
  TotalPages 89
  TotalPagesSent 89
job records
deletejobforall 87
jobsforall 87
reportjoblist 87, 88
returnjoblist 87, 88
JobId 88
JobName 37, 104
jobname 98, 104
jobsforall 87
JobTimeout 37, 38, 43, 102, 107, 109
jobtimeout 98, 105
Jog 5

L
Landscape 77, 79
ledger 98, 115
ledgertray 98, 105, 116
legal 98, 115, 116
legaltray 98, 105, 116, 117
letter 98, 115, 116
lettersmall 98, 115, 116
lettertray 98, 105, 116, 117
LicenseID 43
limitcheck 115, 116
LimitPage 26
LinesPerInch 79
LineWrap 77
LocalTalkType 60, 101
LogicalSize 66, 67, 68, 70, 102, 104, 119

M
MailingTime 22
ManualFeed 5, 79, 105
manualfeed 98, 105
ManualFeedTimeout 8, 102, 105
manualfeedtimeout 98, 105
ManualSize 94
  instance of OutputDevice resource category 94
Margins 8, 106, 109
margins 98, 106
MaxDictStack 37
MaxDisplayList 39
MaxExecStack 37
MaxFaxBuffer 82
MaxFontCache 39
MaxFontItem 37
MaxFormCache 39
MaxFormItem 37
MaxHWRenderingBuffer 43
MaxImageBuffer 44
MaxLJMemory 77, 79
MaxLocalVM 37
MaxOpStack 37
MaxOutlineCache 39
MaxPatternCache 39
MaxPatternItem 37
MaxRasterMemory 44
MaxRetries 22, 82
MaxScreenItem 37
MaxScreenStorage 39
MaxSourceList 44
MaxUPathCache 39
MaxUPathItem 37
MediaColor 5
Index  129

(MediaType 5
MediaWeight 5
MinFontCompress 37
Minute 86
MirrorPrint 5
Month 86
Mounted 66, 68, 69, 71, 72, 85, 119, 120

NegativePrint 5
newsheet 98, 99, 106, 118
NodeID 61
note 98, 115, 116
nPages 28
NumberOfCalls 26
NumCopies 5

On 54, 55, 57, 61, 62, 63, 65, 108
Orientation 5
OutputAttributes 5, 16
OutputDevice 8, 16, 64, 92, 116, 117
OutputDevice resource category instances of
  HWResolution 94
  ManualSize 94
  PageSize 94
OutputFaceUp 5, 106, 109
OutputPage 8
OutputType 5

PageCaption 19, 22, 25, 26, 27
PageCount 41, 45, 76, 106
pagecount 98, 106
Pages 90
PageSize 5, 12, 30, 94, 115, 116, 117
  instance of OutputDevice resource category 94
PageSize Policy 0 116

PostRenderingEnhance 9
PostRenderingEnhanceDetails 9
PostScript Level 2 resources 2
PostScriptPassword 17, 18, 23, 28, 83
PreRenderingEnhance 9
PreRenderingEnhanceDetails 9
PrinterName 19, 45, 60, 106, 109
printername 98, 106
ProcessColorModel 9
processcolors 98, 99, 106
ProcInfo 23
ProcSet 42
product 98, 107
Protocol 54, 57, 58, 64, 110

R
RamSize 45
ramsize 98, 107
rangecheck 32, 86, 100, 116
RealFormat 39, 107
realformat 98, 107
ReceivePostScript 17, 83
RecipientID 23
RecipientMailStop 23
RecipientName 23, 24, 27, 88
RecipientOrg 23, 88
RecipientPhone 23, 27, 88
Regarding 24
RegFontName 80
related publications 3
Removable 68, 71, 72, 120
reportjoblist 87, 88
resources
  Category 42, 93
  ColorRendering 42, 91
  ColorRenderingType 92
  ColorSpace 42, 91
  ColorSpaceFamily 92
Emulator 91, 103
Encoding 42, 91
FaxAdminOps 83, 87
Filter 92
FMapType 92
Font 41, 91
FontType 92
Form 42
FormType 92
Generic 42, 93
Halftone 42, 92
HalftoneType 92
HWOptions 92
ImageType 92
IODevice 47
OutputDevice 92, 116, 117
Pattern 42
PatternType 92
ProcSet 42
RetryInterval 24, 82
returnjoblist 87, 88
RevertToRaster 17, 18, 24, 28, 83
Revision 46, 107
revision 98, 107
Rings 84
Running 86

S
sccbatch 98, 107, 109, 114
sccinteractive 98, 107, 110, 114
Searchable 68, 69, 71, 72, 120
SearchOrder 69, 71, 72, 121
Second 86
SenderID 24
SenderMailStop 24
SenderName 24, 89
SenderOrg 24, 89
SenderPhone 24
SendPostScript 26
SeparationColorNames 10
SeparationOrder 10
Separations 5
ServiceEnable 84
setcolorscreent 37
setdefaulttimeout 98, 107
setdevparams operator 2, 35, 46, 47, 48, 54, 59, 65, 69, 77, 80, 100
setdostartpage 98, 108
setdossystart 98, 108
setduplexmode 98, 118
sethalftone operator 95
sethardwareiomode 98, 108
setjobtimeout 98, 109
setmargins 98, 109
setpagedevice operator 2, 5, 7, 8, 12, 16, 18, 24, 84, 97, 115, 116
setpagestackorder 98, 109
setpapertray 99
setprintername 98, 109
setsccbatch 98, 114
setsccinteractive 98, 115
setsoftwareiomode 98, 110
setsystemparams operator 2, 35, 38, 41, 46, 100
settumble 98, 110, 118
setuserdiskpercent 98, 110
setuserparams operator 2, 35
showpage operator 19, 20
Signature 11
Size 121
SlipSheet 11
softwareiomode 98, 110
Speaker 84
Staple 11
StapleDetails 12
startjob 35, 46, 100, 101
StartJobPassword 35, 36, 46, 100, 101
StartupMode 46, 103, 108
statusdict dictionary 97, 103, 116, 117
compatibility operators in 98
StopBits 59, 112, 114
StorageDevice 82, 84
system parameters
BuildTime 39, 101
ByteOrder 39, 101
CurDisplayList 39
CurFontCache 39
CurFormCache 39
CurInputDevice 39, 110
CurOutlineCache 39
CurOutputDevice 39
CurPatternCache 39
CurScreenStorage 39
CurSourceList 40
CurStoredFontCache 40
CurStoredScreenCache 40
CurUpPathCache 39
DoStartPage 40, 102, 108
FactoryDefaults 36, 40
FatalErrorAddress 41
FontResourceDir 41
GenericResourceDir 41, 42, 43
GenericResourcePathSep 42
JobTimeout 43, 102, 107
LicenseID 43
MaxDiskFontCache 43
MaxDisplayList 39
MaxFontCache 39
MaxFormCache 39
MaxHWRenderingBuffer 43
MaxImageBuffer 44
MaxOutlineCache 39
MaxPatternCache 39
MaxRasterMemory 44
MaxScreenStorage 39
MaxSourceList 44
MaxStoredFontCache 44
MaxStoredScreenCache 45
MaxUpPathCache 39
PageCount 41, 45, 106
Password 100
PrinterName 45, 60, 106, 109
RamSize 45
RealFormat 39, 107
Revision 46, 107
StartJobPassword 35, 46, 100, 101
StartupMode 46, 103, 108
SystemParamsPassword 35, 36, 46, 83, 100, 101, 119, 120
ValidNV 46
WaitTimeout 47, 102, 107
systemdict dictionary 118
compatibility operators in 99
SystemParamsPassword 35, 36, 46, 83, 100, 101, 119, 120

T
TargetID 74
Thresholds 96
TimeBegan 89
timeout 37, 38
TimeSent 26, 90
TimeToStandby 76
TopMargin 78
TotalPages 89
TotalPagesSent 89
TransferFunction 96
TraySwitch 12
Trim 12
TrimWhite 24, 27
tryphone operator 90
Tumble 5, 118
tumble 98, 111, 118
Type 53, 59, 61, 62, 64, 65, 69, 71, 73, 74, 76, 78, 79, 85
type in a Details dictionary 14
type 6 halftone dictionary 95
typeentries
  HalftoneType 96
  Height 96
  Thresholds 96
  TransferFunction 96
  Width 96
typecheck 31

U
user parameters
  AccurateScreens 37
  JobName 37, 104
  JobTimeout 37, 38, 105, 109
  MaxDictStack 37
  MaxExecStack 37
  MaxFontItem 37
  MaxFormItem 37
  MaxLocalVM 37
  MaxPatternItem 37
  MaxScreenItem 37
  MaxUPathItem 37
  MinFontCompress 37
  VMReclaim 37
  VMThreshold 37
  WaitTimeout 38, 111
userdict dictionary 115
  compatibility operators in 98
userdiskpercent 98, 99, 111

V
ValidNV 46
VMI 78
VMReclaim 37
VMThreshold 37

W
WaitForDialTone 85
WaitTimeout 38, 47, 78, 79, 102, 107, 111
waittimeout 98, 111
Width 96
Writeable 69, 71, 73, 85, 102, 120

Y
Year 87