Contents

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 10
2.2 Envelope Orientation in User Space 12
2.3 Errors Generated by Page Device Parameters 13

Chapter 3: Interpreter Parameters 15
3.1 User Parameters 15
3.2 System Parameters 17
3.3 Device Parameters 24

Chapter 4: Resources 43
4.1 Accessing Product Page Device Capability Information 45

Chapter 5: Compatibility 47
5.1 Compatibility Operators 48
5.2 Compatibility Operator Descriptions 49
List of Figures

Figure 2.1 Default user space orientation for portrait PageSize 12
Figure 3.1 Relationship between the communications parameter sets 27
Figure 3.2 Communications parameters sets using NV values 28
Figure 3.3 Communications parameter sets using “hard wired” values 29
# List of Tables

Table 2.1  Page device parameters  6
Table 2.2  Entries in the Policies dictionary  10
Table 3.1  User parameters  16
Table 3.2  System parameters  18
Table 3.3  Parameters common to all devices  30
Table 3.4  Parameters common to device sets of type /FileSystem or /Communications  30
Table 3.5  Parameters common to parameter sets of type /Communications  31
Table 3.6  Parameters common to serial communications parameter sets  32
Table 3.7  Parameters common to LocalTalk communications parameter sets  34
Table 3.8  Parameters common to EtherTalk communications parameter sets  35
Table 3.9  Parameters common to parallel communications parameter sets  36
Table 3.10  Parameters common to /FileSystem devices  36
Table 3.11  Parameters present in cartridge file system devices  39
Table 3.12  Parameters for the LaserJetIIP emulator  40
Table 3.13  Parameters for the color version of the HP7475A plotter emulator  41
Table 3.14  Parameters for the Diablo630 emulator  41
Table 4.1  Regular resources  43
Table 4.2  Resources whose instances are implicit  44
Table 4.3  Resources used in defining new resource categories  44
Table 4.4  Description of keys present in an instance of the category OutputDevice  45
Table 5.1  Stop bits  61
Table 5.2  Data bits  61
Table 5.3  Flow control  61
Table 5.4  Parity  61
Table 5.5  Options byte to devparams conversion  62
Table 5.6  devparams to options byte conversion  63
Table 5.7  Paper size compatibility operators  65
Table 5.8  Paper tray compatibility operators  66
Table 5.9  Attributes for compatibility operator devstatus  69
1.1 Introduction

The purpose of the PostScript Language Supplement for Version 2011 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:

\[
\begin{align*}
\text{currentpagedevice} & \quad \text{setpagedevice} \\
\text{currentuserparams} & \quad \text{setuserparams} \\
\text{currentsystemparams} & \quad \text{setsystemparams} \\
\text{currentdevparams} & \quad \text{setdevparams}
\end{align*}
\]

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 2011 are listed in Chapter 4.

1.1.3 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 5.

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 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) that provides PostScript language programs to be executed by a PostScript product through one of its communication devices.

**PostScript interpreter**: A PostScript interpreter is defined as a piece 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*, Addison-Wesley, Reading, MA, 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.

**Note**  In this list, as well as in Tables 2.1 and 2.2, this symbol is used:
† = this key is present in all 2011 PostScript implementations.

<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>HWResolution†</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></td>
</tr>
<tr>
<td>OutputType</td>
<td>Policies†</td>
<td>OutputFaceUp†</td>
</tr>
<tr>
<td>Tumble</td>
<td></td>
<td>Separations</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><strong>Bind</strong></td>
<td>integer</td>
<td>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>0    Do not bind.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1    Bind at device deactivation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2    Bind at the end of the job.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3    Bind after each set.</td>
</tr>
<tr>
<td><strong>BindDetails</strong></td>
<td>dictionary</td>
<td>This dictionary describes product-specific details related to how a document is to be bound. For more information on <strong>Details</strong> dictionaries, see section 2.1, “Details Dictionaries.”</td>
</tr>
<tr>
<td><strong>Booklet</strong></td>
<td>boolean</td>
<td>Requests that the document be stapled, trimmed, and folded into booklet form.</td>
</tr>
<tr>
<td><strong>BookletDetails</strong></td>
<td>dictionary</td>
<td>This dictionary describes product-specific details related to how a document is to be stapled, trimmed, and folded. For more information on <strong>Details</strong> dictionaries, see section 2.1, “Details Dictionaries.”</td>
</tr>
<tr>
<td><strong>ExitJamRecovery</strong></td>
<td>boolean</td>
<td>If <em>true</em>, pages that jam in the exit path are reprinted. If <em>false</em> (jam recovery disabled), performance might be improved because more overlapping of page processing is possible.</td>
</tr>
<tr>
<td><strong>Fold</strong></td>
<td>integer</td>
<td>Requests that the document be folded. The job will be folded at a specific time indicated by an integer code:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0    Do not fold.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1    Fold at device deactivation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2    Fold at the end of the job.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3    Fold after each set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4    Fold after each <em>showpage</em> or <em>copypage</em>.</td>
</tr>
<tr>
<td><strong>FoldDetails</strong></td>
<td>dictionary</td>
<td>This dictionary describes product-specific details related to how a document is to be folded. For more information on <strong>Details</strong> dictionaries, see section 2.1, “Details Dictionaries.”</td>
</tr>
</tbody>
</table>
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.

Requests that media selection be limited to input sources whose **InputAttributes** include an **InsertSheet** value of *true*. A side-effect of executing `setpagedevice` with **InsertSheet** present and *true* is that the imageable area gets set to a zero-area region to ensure that nothing is imaged on the insert sheet. That is, an 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. It can be inserted into the document between pages \(n\) and \(n+1\) with the intention that it will be affixed to the page at a later time by some other process. Compare with the description of **SlipSheet** below.

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).

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. For more information, see Table 4.11 in section 4.11.3 of the *PostScript Language Reference Manual, Second Edition*.

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.”

**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  
Requests that slip sheets (slip sheet media selection is product specific) be inserted. For example, a slip sheet can be a colored sheet of paper that visually separates multiple copies or a tabbed sheet that separates chapters. A slip sheet *may* be imaged. Compare with the description of `InsertSheet` above.

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

- 0  
   Do not insert slip sheets.
- 1  
   Insert slip sheet at device deactivation.
2 Insert slip sheet at the end of the job.
3 Insert slip sheet at the end of the set.


**Staple** integer Requests that the job be stapled. The job will be stapled at a specific time indicated by an integer code:
0 Do not staple.
1 Staple at device deactivation.
2 Staple at the end of the job.
3 Staple after each set.
4 Staple after each showpage or copypage.


**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 automatically used, without alerting the user that the printer is out of media.

**Trim** integer Requests that the job be trimmed. The job will be trimmed at a specific time indicated by an integer code:
0 Do not trim.
1 Trim at device deactivation.
2 Trim at the end of the job.
3 Trim after each set.
4 Trim after each showpage or copypage.

Table 2.2  Entries in the Policies dictionary

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
</table>
| PageSize | integer | 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:  
7 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.  
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 knowledgable 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 feature 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 applications 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 an integer valued Type entry whose value completely determines how the Details dictionary entries affect 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. Type values and their associated Details dictionaries are registered by Adobe Systems.
2.2 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.

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.

In Figure 2.1, the dashed line indicates that the flap is on the side of the envelope facing down.

*Figure 2.1 Default user space orientation for portrait PageSize*

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.*
2.3 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.

2.3.1 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.
  ```
2.3.2 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.
  ```

2.3.3 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 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 effect 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.

Note In this list, as well as in Table 3.1, these symbols are used:
† = this key is present in all 2011 PostScript implementations.
‡ = this key is present only in all job server (i.e., printer) implementations.

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.

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobName‡</td>
<td>string</td>
<td>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 sent across arbitrary communications channels and is intended for display to users.</td>
</tr>
</tbody>
</table>

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.

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 timeout 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 Indicates the current wait timeout. This value 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: rangecheck, typecheck

3.2 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 effect. 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 this list, as well as in Table 3.2, these symbols are used:
† = this key is present in all 2011 PostScript implementations.
‡ = this key is present only 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>ByteOrder †</td>
<td>integer</td>
<td>(Read-only) A time stamp identifying a specific build of the PostScript interpreter. The values returned by ByteOrder on two different products need not be comparable, and in general, ByteOrder should only be interpreted in conjunction with the manufacturer’s product documentation. Legal values: Any integer. Errors: none</td>
</tr>
<tr>
<td>CurDisplayList †</td>
<td>string</td>
<td>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 %Serial% devices, see section 3.2.2. Legal values: A string containing a communications device name. Errors: none</td>
</tr>
<tr>
<td>CurFontCache †</td>
<td>string</td>
<td>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 %Serial% devices, see section 3.2.2. Legal values: A string containing a communications device name. Errors: none</td>
</tr>
<tr>
<td>CurOutlineCache †</td>
<td>string</td>
<td>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 %Serial% devices, see section 3.2.2. Legal values: A string containing a communications device name. Errors: none</td>
</tr>
<tr>
<td>CurPatternCache †</td>
<td>string</td>
<td>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 %Serial% devices, see section 3.2.2. Legal values: A string containing a communications device name. Errors: none</td>
</tr>
<tr>
<td>CurScreenStorage †</td>
<td>string</td>
<td>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 %Serial% devices, see section 3.2.2. Legal values: A string containing a communications device name. Errors: none</td>
</tr>
<tr>
<td>CurUPathCache †</td>
<td>string</td>
<td>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 %Serial% devices, see section 3.2.2. Legal values: A string containing a communications device name. Errors: none</td>
</tr>
<tr>
<td>MaxDisplayList †</td>
<td>integer</td>
<td>(Read-only) Indicates the maximum number of display lists that can be in storage simultaneously. Legal values: Any integer. Errors: none</td>
</tr>
<tr>
<td>MaxFontCache †</td>
<td>integer</td>
<td>(Read-only) Indicates the maximum number of fonts that can be in storage simultaneously. Legal values: Any integer. Errors: none</td>
</tr>
<tr>
<td>MaxFormCache †</td>
<td>integer</td>
<td>(Read-only) Indicates the maximum number of form objects that can be in storage simultaneously. Legal values: Any integer. Errors: none</td>
</tr>
<tr>
<td>MaxOutlineCache †</td>
<td>integer</td>
<td>(Read-only) Indicates the maximum number of outline objects that can be in storage simultaneously. Legal values: Any integer. Errors: none</td>
</tr>
<tr>
<td>MaxScreenStorage †</td>
<td>integer</td>
<td>(Read-only) Indicates the maximum number of screen operators that can be in storage simultaneously. Legal values: Any integer. Errors: none</td>
</tr>
<tr>
<td>MaxPatternCache †</td>
<td>integer</td>
<td>(Read-only) Indicates the maximum number of pattern operators that can be in storage simultaneously. Legal values: Any integer. Errors: none</td>
</tr>
<tr>
<td>MaxUPathCache †</td>
<td>integer</td>
<td>(Read-only) Indicates the maximum number of UPath objects that can be in storage simultaneously. Legal values: Any integer. Errors: none</td>
</tr>
<tr>
<td>RealFormat †</td>
<td>integer</td>
<td>(Read-only) Indicates the current format in use. Legal values: Any integer. Errors: none</td>
</tr>
<tr>
<td>CurInputDevice ‡</td>
<td>string</td>
<td>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 %Serial% devices, see section 3.2.2. Legal values: A string containing a communications device name. Errors: none</td>
</tr>
<tr>
<td>StartJobPassword ‡</td>
<td>string</td>
<td>Indicates the password used to start a job. Legal values: A string containing a communications device name. Errors: none</td>
</tr>
<tr>
<td>SystemParamsPassword ‡</td>
<td>string</td>
<td>Indicates the password used to access system parameters. Legal values: A string containing a communications device name. Errors: none</td>
</tr>
</tbody>
</table>
```

Table 3.2 describes system parameters that have been defined or amended since the publication of the PostScript Language Reference Manual, Second Edition. The following are amended system parameters:

**3.2.1 System Parameter Table**

Table 3.2 System parameters

<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>StartJobPassword ‡</td>
<td>string</td>
<td>Indicates the password used to start a job. Legal values: A string containing a communications device name. Errors: none</td>
</tr>
<tr>
<td>SystemParamsPassword ‡</td>
<td>string</td>
<td>Indicates the password used to access system parameters. Legal values: A string containing a communications device name. Errors: none</td>
</tr>
</tbody>
</table>
CurOutputDevice†

string (Read-only) 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 communications device parameter set whose values are normally stored in RAM: for example, (%Serial%). For more information on %Serial% devices, see section 3.2.2.

Legal values: A string containing a communications device name.

Errors: none

CurSourceList

integer (Read-only) 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

DoStartPage

boolean Indicates whether the start page should print when the device is powered on. The start page prints if the value of DoStartPage is true at power-on.

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**

integer

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 sent 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 at power-on or device restart, the address is printed on the start page or possibly on a separate page.

**Legal values:** Any integer.

**Errors:** none

**FontResourceDir** †

string

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/). 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 Manual, Second Edition, section 3.9, “Named Resources.”) 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

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), **GenericResourceDir** should be set to `%null`. This mechanism can also be used by site administrators to temporarily disable access to external resources.

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

**Errors:** limitcheck, typecheck

**JobTimeout** integer Indicates the value in seconds to which the user parameter **JobTimeout** will be initialized at the beginning of each job. A value of 0 indicates that the timeout will be infinite. The reason for the minimum value of 15 is that small values may prevent a job from setting **JobTimeout** to another value successfully.

**Legal values:** 0 or any integer greater than or equal to 15.

**Errors:** rangecheck, typecheck

**LicenseID** string This parameter contains the Adobe-assigned license identifier. Its value is unique to each product.

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

**Errors:** limitcheck, typecheck
MaxRasterMemory

integer

Indicates the largest amount of memory 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 PostScript 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 at boot time; any changes to the value of the parameter will not take effect until then.

Legal values: Product dependent.

Errors: typecheck

MaxSourceList

integer

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

PageCount

integer

(Read-only) 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 but will reset in the event that the non-volatile memory in which the parameter is stored has been corrupted.

Legal values: Any non-negative integer.

Errors: none

PrinterName

string

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. If string is a zero length string, PrinterName is 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)* 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)* 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 each time setsystemsparams, setdevparams, or other system administrator operations are invoked.

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

**Errors:** limitcheck, typecheck

StartupMode  integer  Indicates whether the system start file (Sys/Start) or some other start-up procedure should be executed when the device is powered on. The Sys/Start file executes if the value of StartupMode is 1 at power-on. If the StartupMode value is 0, no special start-up procedures are run at power-on. 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:** rangecheck, 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 setsystemsparams, 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) Indicates whether non-volatile memory is currently used to store persistent parameters. At power-on, 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 at power-off and factory defaults are used at power-on.

Legal values: true, false

Errors: none

WaitTimeout integer 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. The reason for the minimum value of 15 is that small values may prevent a job from setting WaitTimeout to another value successfully.

Legal values: 0 or any integer greater than or equal to 15.

Errors: rangecheck, typecheck

3.3 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, 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.). Each named device known to the currentdevparams/setdevparams operators corresponds to an instance of the IODevice resource category and typically represents a set of parameters.
describing the configuration of a physical or logical communication channel. (See section C.4 of the Postscript Language Reference Manual, Second Edition for more details.) More generally, “device” in this context really means “named parameter set.” 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.

3.3.1 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 most communication device sets there is an On and an Enabled parameter. The On parameter determines whether that communications channel has a device driver turned on so that data sent from a host will be properly received. The Enabled parameter determines whether data received on the channel is to be considered for the queue of PostScript language jobs to be executed. A channel that is On but not Enabled might contain pure data to be directly read by an executing PostScript language job. As such, it should not be put on the queue as a separate PostScript language job. A channel that is Enabled but not On does not make sense, however, and as such, it is 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. If both LocalTalk and a serial channel share the same hardware port (as is often the case), 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.

3.3.2 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, the parallel port is associated with the parameter device set named %Parallel%. Channel A of the serial communications controller (SCC) chip, which normally is wired to a 25-pin connector, is associated with the parameter device set named %Serial%. Channel B of the SCC chip, which normally 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. Most commonly, the %SerialB% device and the %LocalTalk% device are both 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:

\[
\begin{align*}
%Serial_NV% & \quad %Serial% & \quad %Serial_Pending% \\
%SerialB_NV% & \quad %SerialB% & \quad %SerialB_Pending% \\
%Parallel_NV% & \quad %Parallel% & \quad %Parallel_Pending% \\
%LocalTalk_NV% & \quad %LocalTalk% & \quad %LocalTalk_Pending% 
\end{align*}
\]

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 and survive across printer restarts.
- %CommName% values usually are stored in RAM and survive until the next printer restart.
- %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.

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 drivers need not take these differences into account. If their specific intent is to affect values across
power cycles, 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**

![Diagram showing the relationship between the communications parameter sets]

Figure 3.1 shows the basic relationship of the three parameter sets.
In this figure, values written to \%CommName\% are written through to
\%CommName_NV\%; and values written to \%CommName_NV\% are
written through to \%CommName\% and thence to
\%CommName_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 \%CommName\% or to \%CommName_NV\% if
  the value is to persist across power cycles. Some products store to only
  one of these sets.

- The product may have switches through which it can be directed to use
  either \%CommName_NV\% parameter sets or built-in (hard-wired)
  values. (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 \%CommName\% or \%CommName_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 \%CommName\% parameter set, which is in RAM and does not survive across restarts, is used in many cases (but not all) to update the \%CommName\_Pending\% set. Thus, on many products (those with a front panel but no switches), the \%CommName\% and \%CommName\_Pending\% sets always have the same values and appear redundant.

The \%CommName\_NV\% set usually holds values across printer restarts. 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 be saved across restarts.

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 in the case with a single non-volatile set, these parameter sets obtain their values by being written to by a PostScript spooler or utility.

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

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 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\%. Resetting 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 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 be saved across restarts.

### 3.3.3 Device Parameter Tables

Tables 3.3 through 3.14 list all device parameters that are currently defined.

#### Table 3.3 Parameters common to all devices

<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) Describes 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, /Fax, /FileSystem,

**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) Indicates whether the device represented by the parameter set supports named files. This is defined only in device parameter sets of the type /FileSystem or /Communications.</td>
</tr>
</tbody>
</table>

**Legal values:** true, false

**Errors:** none
### Table 3.5 Parameters common to parameter sets of type /Communications

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
</table>
| Enabled | boolean | Determines 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.  
- In the same parameter set, **On** is **false** and **Enabled** is **true**.  
- All communication devices have **Enabled** set to **false**.  
**Legal values:** true, false  
**Errors:** configurationerror, typecheck |
| Interpreter | name | Determines the type of executable job the arriving data represents. This parameter is used only if **Enabled** is **true**. For certain devices, there is a relationship between the **Interpreter** and the **Protocol** parameters that can result in a **configurationerror**. See **Protocol** for details.  
Either **Interpreter** or **Protocol** or both can be set without a password if no other parameters are specified in the execution of **setdevparams**.  
For more information on the legal values listed below, see section 3.3.5 “Emulator Parameters.”  
**Legal values:** /Diablo630, /HP7475A, /LaserJetIIP, /PostScript, /ProprinterXL  
**Errors:** configurationerror, rangecheck, typecheck |
| On      | boolean | Determines whether the device driver for the communications device is turned on and able to receive and send data. If the parameter is **true**, data sent 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 common to 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>Determines 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. <strong>Legal values:</strong>  Product dependent. <strong>Errors:</strong>  rangecheck, typecheck</td>
</tr>
<tr>
<td>CheckParity</td>
<td>boolean</td>
<td>Determines 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. <strong>Legal values:</strong>  true, false <strong>Errors:</strong>  typecheck</td>
</tr>
<tr>
<td>DataBits</td>
<td>integer</td>
<td>Determines 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 sent or received is the sum of the number of start bits (always 1), data bits, parity bits, and stop bits. <strong>Legal values:</strong>  7, 8 <strong>Errors:</strong>  rangecheck, typecheck</td>
</tr>
<tr>
<td>FlowControl</td>
<td>name</td>
<td>Determines the serial flow control method used between the host and the device. 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 sent. A high value for the signal indicates that data may be sent, a low value indicates that data should not be sent.</td>
</tr>
</tbody>
</table>
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 sent and a high signal indicates that data should not be sent.

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: The same as 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. 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.

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

Errors: rangecheck, typecheck

Parity name Determines 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. The total number of bits for each byte sent 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 name Indicates the communications protocol that will be used. These communications protocols are available:

Binary: In /Binary mode, the full range of 8-bit data can be sent. Emulators and the PostScript interpreter can use this mode.

Normal: In /Normal mode, certain control characters have special meanings and indicate something about the job being received. /Normal mode is used only with the PostScript interpreter.

Raw: In /Raw mode, everything received by the device driver is data to be passed on, unchanged, to the printer emulator. The /Raw mode is used only by a printer emulator, not by the PostScript interpreter.
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*

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. Any job can be executed when the protocol is */Binary*.

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*

**Errors:** *configurationerror, rangecheck, typecheck*

**StopBits** integer Determines the number of stop bits that will be sent by the serial hardware. The hardware will always be able to receive data sent with one stop bit. The total number of bits for each byte sent 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*

<table>
<thead>
<tr>
<th><strong>Table 3.7 Parameters common to LocalTalk communications parameter sets</strong></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LocalTalkType</strong></td>
<td>string</td>
<td>Represents the <em>type</em> piece of the AppleTalk <em>entity name</em>. 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 <strong>PrinterName</strong> system parameter and the zone is wildcarded.</td>
</tr>
<tr>
<td><strong>Legal values:</strong></td>
<td>Any string of 32 or fewer non-null characters.</td>
<td></td>
</tr>
<tr>
<td><strong>Errors:</strong></td>
<td><em>limitcheck, typecheck</em></td>
<td></td>
</tr>
</tbody>
</table>
### NodeID

**integer**  
*(Read-only)* 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

---

**Table 3.8** Parameters common to EtherTalk communications parameter sets

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>EthernetAddress</td>
<td>string</td>
<td><em>(Read-only)</em> 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.</td>
</tr>
<tr>
<td><strong>Legal Values:</strong></td>
<td></td>
<td>Any string of 17 characters.</td>
</tr>
<tr>
<td><strong>Errors:</strong></td>
<td></td>
<td>none</td>
</tr>
<tr>
<td>EtherTalkType</td>
<td>string</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Legal Values:</strong></td>
<td></td>
<td>Any string of 32 or fewer non-null characters.</td>
</tr>
<tr>
<td><strong>Errors:</strong></td>
<td></td>
<td>typecheck</td>
</tr>
<tr>
<td>EtherTalkZone</td>
<td>string</td>
<td>Represents the zone piece of the EtherTalk entity name.</td>
</tr>
<tr>
<td><strong>Legal Values:</strong></td>
<td></td>
<td>Any string of 32 or fewer non-null characters.</td>
</tr>
<tr>
<td><strong>Errors:</strong></td>
<td></td>
<td>typecheck</td>
</tr>
</tbody>
</table>
Table 3.9 Parameters common to parallel communications parameter sets

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>name</td>
<td>This parameter has the same interpretation as the corresponding serial parameter of the same name.</td>
</tr>
</tbody>
</table>

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

**Errors:** configurationerror, rangecheck, typecheck

3.3.4 Disk and Cartridge Parameter Tables

The following is 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.

In Table 3.10, it should be understood that a page is a unit of storage whose size is file-system dependent.

Table 3.10 Parameters common to /FileSystem devices

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free</td>
<td>integer</td>
<td><em>(Read-only)</em> Indicates the amount of free space (in pages for disks and bytes for cartridges) on the media for the device.</td>
</tr>
</tbody>
</table>

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

**Errors:** none

| HasNames | boolean | *(Read-only)* Indicates whether the device supports named files. If false, the device is a single entity of data. |

**Legal values:** true, false

**Errors:** none
**InitializeAction**

integer 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. For more information, see **LogicalSize**.

2 reformats the entire media before creating a new file system of size **LogicalSize**.

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**.

The following are valid values for cartridges:

0 indicates no action and is the value returned when the parameter is read.

1 reformats the entire media before creating a new file system of size **PhysicalSize**.

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

**Errors:** none

**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 device. For more information, see **InitializeAction**.

When queried, this parameter indicates the current size (in pages for disks and bytes for cartridges) of the file system on the device. A value of 0 indicates that there is no valid file system.

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 device is reformatted, it may return a different value from what was set because it will return the current size of the device.

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

**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.

Legal values:  true, false

Errors:  typecheck

PhysicalSize  integer   (Read-only) Indicates the size (in pages for disks and bytes for cartridges) of the media.

Legal values:  Any non-negative integer.

Errors:  none

Removable  boolean   (Read-only) Indicates whether the media of the device can be removed.

Legal values:  true, false

Errors:  none

Searchable  boolean   (Read-only) Indicates whether the device participates in searches in operations that require a device but for which no device has been specified.

Legal values:  true, false

Errors:  none

SearchOrder  integer   (Read-only) Indicates the priority at which the device participates when searching for a file in operations in which no device has been specified. 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.

Legal values:  /FileSystem

Errors:  none
Writeable  boolean  *(Read-only)* Indicates whether the device currently can be written to. Usually this boolean is *true* only if the media is physically writeable and not write-protected.

**Legal values:**  true, false  

**Errors:**  none 

---

**Table 3.11** Parameters present in cartridge file system devices

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>CartridgeID</td>
<td>integer</td>
<td><em>(Read-only)</em> 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.</td>
</tr>
<tr>
<td>CartridgeType</td>
<td>integer</td>
<td><em>(Read-only)</em> Indicates the category classification of the cartridge. This classification is a registry maintained by Adobe Systems. Currently the only category is 4, which indicates a font cartridge.</td>
</tr>
</tbody>
</table>

**Legal values:**  Any integer.  

**Errors:**  none 

---

### 3.3.5 Emulator Parameters

An emulator is an alternative interpreter for the input stream. Some PostScript printers have the ability to emulate other printers. The **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 **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:

```
(%SerialA%)<</Interpreter /Diablo630
/Protocol /Raw>> setdevparams
```
gives Diablo630-like functionality to input channel SerialA on a PostScript printer that has a Diablo630 emulator.

The Diablo630 emulator, the color version of the HP7475A plotter emulator, and the LaserJetIIP 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.

The following tables describe the parameters for the LaserJetIIP emulator, the color version of the HP7475A plotter emulator, and the Diablo630 emulator.

**Table 3.12 Parameters for the LaserJetIIP emulator**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copies</td>
<td>integer</td>
<td>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>Specifies the desired font height in points.</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 FontFixed is true. FontPitch takes a real number specifying the number of characters per inch.</td>
</tr>
<tr>
<td>FontSymbolSet</td>
<td>integer</td>
<td>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.</td>
</tr>
<tr>
<td>FontTypeface</td>
<td>integer</td>
<td>The value of FontTypeface is the number assigned to a particular font.</td>
</tr>
<tr>
<td>FontWeight</td>
<td>integer</td>
<td>Specifies the “weight” or “boldness” of desired font.</td>
</tr>
<tr>
<td>Landscape</td>
<td>boolean</td>
<td>If true, the initial orientation of the page will be landscape instead of portrait.</td>
</tr>
<tr>
<td>LinesPerInch</td>
<td>real</td>
<td>Specifies the default value for the “vertical motion index.” This determines the interline spacing (and hence the number of lines on the page).</td>
</tr>
</tbody>
</table>
**MaxLJMemory** integer  
This parameter allows the user to limit the amount of memory that the LaserJetIIP emulator will take for its needs. Without this limit, the emulator will acquire memory at the expense of other needs, such as VM and font cache. Within a given emulation job, the LaserJetIIP emulator will use temporary memory in excess of **MaxLJMemory** to hold fonts and macros.

**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 3.13 Parameters for the color version of the HP7475A plotter emulator**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ColorSetup</strong></td>
<td>string</td>
<td>This parameter allows the user to change the default pen color. The <strong>ColorSetup</strong> 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).</td>
</tr>
</tbody>
</table>

---

**Table 3.14 Parameters for the Diablo630 emulator**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AutoLF</strong></td>
<td>boolean</td>
<td>If <em>true</em>, automatic line feeding is specified.</td>
</tr>
<tr>
<td><strong>BoldFontName</strong></td>
<td>name</td>
<td>Specifies the name of the PostScript font used for boldface printing when <strong>ECS</strong> is <em>false</em>.</td>
</tr>
<tr>
<td><strong>ECS</strong></td>
<td>boolean</td>
<td>If <em>true</em>, the printer emulates the IBM PC Graphics ECS (extended character set) print wheel. If <em>false</em>, the printer emulates the 96-character plastic print wheel.</td>
</tr>
<tr>
<td><strong>ECSDataWidth</strong></td>
<td>integer</td>
<td>Selects 7- or 8-bit data when <strong>ECS</strong> is <em>true</em>.</td>
</tr>
<tr>
<td><strong>RegFontName</strong></td>
<td>name</td>
<td>Specifies the name of the PostScript font used for regular printing when <strong>ECS</strong> is <em>false</em>.</td>
</tr>
</tbody>
</table>
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 present in all implementations of PostScript language version 2011.

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>
<tr>
<td>Font</td>
<td>Courier Courier-Bold</td>
</tr>
<tr>
<td></td>
<td>Courier-BoldOblique Courier-Oblique</td>
</tr>
<tr>
<td></td>
<td>Helvetica Helvetica-Bold</td>
</tr>
<tr>
<td></td>
<td>Helvetica-BoldOblique Helvetica-Oblique</td>
</tr>
<tr>
<td></td>
<td>Symbol Times-Bold</td>
</tr>
<tr>
<td></td>
<td>Times-BoldItalic Times-Italic</td>
</tr>
<tr>
<td></td>
<td>Times-Roman</td>
</tr>
<tr>
<td></td>
<td>ZapfChancery-MediumItalic</td>
</tr>
<tr>
<td></td>
<td>ZapfDingbats</td>
</tr>
<tr>
<td>Halftone</td>
<td>No instances defined.</td>
</tr>
<tr>
<td>OutputDevice</td>
<td>Default</td>
</tr>
</tbody>
</table>
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</td>
</tr>
<tr>
<td></td>
<td>CIEBasedABC</td>
</tr>
<tr>
<td></td>
<td>DeviceCMYK</td>
</tr>
<tr>
<td></td>
<td>DeviceGray</td>
</tr>
<tr>
<td></td>
<td>DeviceRGB</td>
</tr>
<tr>
<td></td>
<td>Indexed</td>
</tr>
<tr>
<td>Pattern</td>
<td>Separation</td>
</tr>
<tr>
<td>Filter</td>
<td>ASCII85Decode</td>
</tr>
<tr>
<td></td>
<td>ASCII85Encode</td>
</tr>
<tr>
<td></td>
<td>ASCIIHexDecode</td>
</tr>
<tr>
<td></td>
<td>ASCIIHexEncode</td>
</tr>
<tr>
<td></td>
<td>CCITTFaxDecode</td>
</tr>
<tr>
<td></td>
<td>CCITTFaxEncode</td>
</tr>
<tr>
<td></td>
<td>DCTDecode</td>
</tr>
<tr>
<td></td>
<td>DCTEncode</td>
</tr>
<tr>
<td></td>
<td>LZWDecode</td>
</tr>
<tr>
<td></td>
<td>LZWEncode</td>
</tr>
<tr>
<td></td>
<td>NullEncode</td>
</tr>
<tr>
<td></td>
<td>RunLengthDecode</td>
</tr>
<tr>
<td></td>
<td>RunLengthEncode</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</td>
</tr>
<tr>
<td>ImageType</td>
<td>1</td>
</tr>
<tr>
<td>PatternType</td>
<td>1</td>
</tr>
</tbody>
</table>

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>Category</td>
<td>ColorRendering</td>
</tr>
<tr>
<td>ColorSpace</td>
<td>ColorRenderingType</td>
</tr>
<tr>
<td>Encoding</td>
<td>Filter</td>
</tr>
<tr>
<td>Font</td>
<td>FontType</td>
</tr>
<tr>
<td>Generic</td>
<td>HalftoneType</td>
</tr>
<tr>
<td>IODevice</td>
<td>OutputDevice</td>
</tr>
<tr>
<td>PatternType</td>
<td>ProcSet</td>
</tr>
</tbody>
</table>
4.1 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 2011 products, the entries listed in Table 4.4 are present:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HWResolution</code></td>
<td>Array of <code>HWResolution</code> values which can be supported by the product.</td>
</tr>
</tbody>
</table>

Each element can be either an array of two numbers indicating a discrete `HWResolution` 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.

| ManualSize | Array of page sizes for the product which can be fed manually. |

Each element can be either an array of two numbers indicating a discrete 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.
PageSize  Array of page sizes for the product which can be fed automatically (assuming appropriate media are installed.

Each element can be either an array of two numbers indicating a discrete 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.
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 that were often present in PostScript Level 1 products, a collection of statusdict operators and procedures 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 in PostScript Level 2 drivers because the presence or absence of the operators is product dependent. Instead, the appropriate Level 2 standard operators should be used.
5.1 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 present in all 2011 PostScript implementations unconditionally.
‡ = this compatibility operator is present in all 2011 PostScript implementations unconditionally. However, in the absence of the associated feature, it performs no function aside from its documented effect on the operand stack.
* = this compatibility operator is present in the dictionary userdict.
§ = 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>a3tray</th>
<th>a4tray</th>
<th>applletalktype</th>
</tr>
</thead>
<tbody>
<tr>
<td>b5tray</td>
<td>buildtime †</td>
<td>byteorder †</td>
</tr>
<tr>
<td>checkpassword †</td>
<td>defaulttimeout ‡</td>
<td>diskonline</td>
</tr>
<tr>
<td>diskstatus</td>
<td>dostartpage</td>
<td>dosysstart</td>
</tr>
<tr>
<td>duplexmode</td>
<td>emulate</td>
<td>firstside</td>
</tr>
<tr>
<td>hardwareiomode ‡</td>
<td>initializedisk</td>
<td>jobname</td>
</tr>
<tr>
<td>jobtimeout</td>
<td>ledgertray</td>
<td>legaltray</td>
</tr>
<tr>
<td>lettertray</td>
<td>manualfeed</td>
<td>manualfeedtimeout</td>
</tr>
<tr>
<td>margins</td>
<td>newsheet</td>
<td>pagecount ‡</td>
</tr>
<tr>
<td>pagemastorder</td>
<td>prinntername</td>
<td>product †</td>
</tr>
<tr>
<td>ramsize</td>
<td>realformat †</td>
<td>revision †</td>
</tr>
<tr>
<td>sccbatch</td>
<td>sccinteractive ‡</td>
<td>setdefaulttimeout §‡</td>
</tr>
<tr>
<td>setdostartpage §</td>
<td>setdosysstart §</td>
<td>setduplexmode</td>
</tr>
<tr>
<td>sethardwareiomode §§</td>
<td>setjobtimeout ‡</td>
<td>setmargins §§</td>
</tr>
<tr>
<td>setpagemastorder ‡</td>
<td>setprinntername §§</td>
<td>setscrbatch §</td>
</tr>
<tr>
<td>setscmbinteractive</td>
<td>setsoftwareiomode §§</td>
<td>stumble</td>
</tr>
<tr>
<td>setuserdiskpercent</td>
<td>softwareiomode ‡</td>
<td>tumble</td>
</tr>
<tr>
<td>userdiskpercent</td>
<td>waittimeout ‡</td>
<td>11x17tray</td>
</tr>
</tbody>
</table>

In **userdict**:

<table>
<thead>
<tr>
<th>a3 *</th>
<th>a4 *</th>
<th>a4small *</th>
</tr>
</thead>
<tbody>
<tr>
<td>b5 *</td>
<td>ledger *</td>
<td>legal * ‡</td>
</tr>
<tr>
<td>letter * ‡</td>
<td>lettersmall *</td>
<td>note *</td>
</tr>
<tr>
<td>11x17 *</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In `systemdict`:

- `devdismount`†
- `devforall`†
- `devformat`†
- `devmount`†
- `devstatus`†

### 5.2 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 every Level 1 compatibility operation. Thus, the 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`.

#### 5.2.1 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`.

#### 5.2.2 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 compatibility 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 `setsystemsparams` 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 unencapsulated 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 but 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.

**a3tray**

See section 5.2.5.

**a4tray**

See section 5.2.5.

**appletalktype**

- **appletalktype** string

is a string with the same value as the LocalTalkType device parameter in the `%LocalTalk%` parameter set. Redefining `appletalktype` redefines the `LocalTalkType` parameter as well as the EtherTalkType parameter in the `%EtherTalk%` parameter set. However, changes to the EtherTalkType parameter do not change the string returned by the `appletalktype` operator.

The compatibility operator `appletalktype` is present only if either the `%LocalTalk%` or `%EtherTalk%` device names are present.

**Errors:** stackoverflow
b5tray

See section 5.2.5.

buildtime†

– buildtime int

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

Errors: stackoverflow

byteorder†

– byteorder bool

is a boolean with the same value as the system parameter ByteOrder.

Errors: stackoverflow

checkpassword†

int checkpassword bool

string checkpassword bool

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

Errors: stackunderflow, typecheck

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 registered as present and operational. This is determined by searching all device parameter sets named %diskN%, where N is any integer. 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 %diskN%, where N is any integer, that have a **Writeable** parameter set to true. 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**

- **dosysstart bool**

returns true if and only if the value of the system parameter **StartupMode** is 1.

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

**Errors:** stackoverflow

**duplexmode**

See section 5.2.6.
emulate

\textit{input-stream emulation-name emulate} –
or
\textit{input-stream params-dict emulation-name emulate} –

Causes the PostScript language interpreter to yield control, and the emulator named by \textit{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 \textit{emulation-name} may be found in the implicit resource category \texttt{Emulator}. An illegal \textit{emulation-name} will cause a \texttt{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 \textit{input-stream} is a file object which becomes the input source for the emulator. The \textit{input-stream} specified must be appropriate to the product-dependent emulator, as defined in the product Addendum. An illegal \textit{input-stream} will cause an \texttt{invalidaccess} error.

\textbf{Legal Values:} Described above.

\textbf{Errors:} \texttt{invalidaccess, rangecheck, stackoverflow, stackunderflow},

\texttt{firstside} See section 5.2.6.
hardwareiomode ‡ – hardwareiomode int

returns int which indicates a current communications channel whose corresponding device parameter set Enabled boolean is true. Because multiple channels may be enabled, the smallest such int is returned. The interpretation of int is:

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

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
legaltray
lettertray

manualfeed

manualfeed

manualfeed

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. The operator manualfeed does not set or return the value of ManualFeed.

The compatibility operator manualfeed 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

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. The operator manualfeedtimeout does not set or return the value of ManualFeedTimeout.

Errors: stackoverflow

margins

margins

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

Errors: stackoverflow
newsheet

See section 5.2.6.

pagecount

– pagecount int

returns the value of the system parameter PageCount.

Errors: stackoverflow

pagestackorder

– pagestackorder bool

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

string prinename substring

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

Errors: rangecheck, stackunderflow, typecheck

product

– product string

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

Errors: stackoverflow

ramsize

– ramsise int

returns the number of bytes of RAM available to the product.

Errors: stackoverflow

realformat

– realformat string

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

Errors: stackoverflow
revision\(^\dagger\)  –  revision  \textit{int}  

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

\textbf{Errors:}  
stackoverflow

\textbf{sccbatch}  
See section 5.2.3.

\textbf{sccinteractive}\(^\dagger\)  
See section 5.2.3.

\textbf{setdefaulttimeouts} \(^\S\S\)  
\textit{job} manufeed \textit{wait} setdefaulttimeouts  –  

sets the system parameters \textit{JobTimeout} and \textit{WaitTimeout} to \textit{job} and \textit{wait}, respectively, and sets the page device parameter \textit{ManualFeedTimeout} to \textit{manufeed}. \textit{setdefaulttimeouts} always takes three values, even if the corresponding system parameters are not present.

\textbf{Errors:}  
invalidaccess, rangecheck, stackunderflow, typecheck

\textbf{setdostartpage} \(^\S\)  
\textit{bool} setdostartpage  –  

sets the system parameter \textit{DoStartPage} to the value of \textit{bool}.

The system parameter \textit{DoStartPage} must be present for the compatibility operator \textit{setdostartpage} to be present.

\textbf{Errors:}  
invalidaccess, stackunderflow, typecheck

\textbf{setdosysstart} \(^\S\)  
\textit{bool} setdosysstart  –  

sets the system parameter \textit{StartupMode} according to the value of \textit{bool}. \textit{StartupMode} is set to 1 if \textit{bool} is \textit{true} and set to 0 if \textit{bool} is \textit{false}.

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

\textbf{Errors:}  
invalidaccess, stackunderflow, typecheck
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:

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

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

Errors: invalidaccess, rangecheck, stackunderflow, typecheck

int setjobtimeout

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

top left setmargins

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: rangecheck, stackunderflow, typecheck

bool setpagestackorder

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: stackunderflow, typecheck
setprintername

string setprintername –

sets the system parameter PrinterName to the value of string.

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

Errors: invalidaccess, limitcheck, stackunderflow, typecheck

setsccbatch

See section 5.2.3.

setscctinteractive

See section 5.2.3.

setsoftwareiomode

int setsoftwareiomode –

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:

0  Interpreter: PostScript;  Protocol: Normal
1  Interpreter: ProprinterXL;  Protocol: Raw
2  Interpreter: Diablo630;  Protocol: Raw
3  Interpreter: TI855;  Protocol: Raw
4  Interpreter: HP7475A;  Protocol: Raw
5  Interpreter: LaserJetIIP;  Protocol: Raw
100  Interpreter: PostScript;  Protocol: Binary

Errors: invalidaccess, rangecheck, stackunderflow, typecheck

settumble

See section 5.2.6.

setuserdiskpercent

int setuserdiskpercent –

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

Errors: rangecheck, stackunderflow, typecheck
softwareiomode ‡ – \texttt{softwareiomode \ int}

returns \texttt{int} which indicates (see \texttt{setsoftwareiomode}) the interpretation mode for the current communications device (as indicated by the system parameter \texttt{CurInputDevice}).

The \texttt{Serial}, \texttt{Parallel}, \texttt{SerialB}, or \texttt{LocalTalk} device parameter set must be present for the compatibility operator \texttt{setsoftwareiomode} and \texttt{softwareiomode} to be present.

Errors: stackoverflow

tumble

See section 5.2.6.

userdiskpercent – \texttt{userdiskpercent \ int}

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

Errors: stackoverflow

waittimeout ‡ – \texttt{waittimeout \ int}

is an integer with the same value as the user parameter \texttt{WaitTimeout}. Redefining either \texttt{waittimeout} or the user parameter \texttt{WaitTimeout} redefines the other to the same value.

The user parameter \texttt{WaitTimeout} must be present for the compatibility operator \texttt{waittimeout} to be present.

Errors: stackoverflow

11x17tray

See section 5.2.5.
5.2.3 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 5.1 Stop bits

<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 5.2 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 5.3 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 5.4 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. The tables below 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 5.5 Options byte to devparams conversion**

<table>
<thead>
<tr>
<th>data bits &amp; parity --&gt; DataBits &amp; Parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>standard space</td>
</tr>
<tr>
<td>standard mark</td>
</tr>
<tr>
<td>standard odd</td>
</tr>
<tr>
<td>standard even</td>
</tr>
<tr>
<td>7 bits space</td>
</tr>
<tr>
<td>7 bits mark</td>
</tr>
<tr>
<td>7 bits odd</td>
</tr>
<tr>
<td>7 bits even</td>
</tr>
<tr>
<td>8 bits space</td>
</tr>
<tr>
<td>8 bits mark</td>
</tr>
<tr>
<td>8 bits odd</td>
</tr>
<tr>
<td>8 bits even</td>
</tr>
</tbody>
</table>
Table 5.6  devparams to options byte conversion

<table>
<thead>
<tr>
<th>DataBits &amp; Parity --&gt; data bits &amp; parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 bits None</td>
</tr>
<tr>
<td>7 bits Space</td>
</tr>
<tr>
<td>7 bits Mark</td>
</tr>
<tr>
<td>7 bits Odd</td>
</tr>
<tr>
<td>7 bits Even</td>
</tr>
<tr>
<td>8 bits None</td>
</tr>
<tr>
<td>8 bits Space</td>
</tr>
<tr>
<td>8 bits Mark</td>
</tr>
<tr>
<td>8 bits Odd</td>
</tr>
<tr>
<td>8 bits Even</td>
</tr>
</tbody>
</table>

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.

SCC Compatibility Operators

```
channel sccbatch baud options
```

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 5.6. The values for baud, stop bits, and flow control are determined from the corresponding settings for the Baud, StopBits, and FlowControl device parameters, respectively.

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 \(\dagger\)

```
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:** rangecheck, stackoverflow, stackunderflow, typecheck

**setscbbatch \(\S\)**

```
channel  baud  options  setscbbatch  –
```

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``` and ```stop bits``` and ```flow control``` respectively. ```DataBits``` and ```Parity``` are set based on Table 5.6 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 ```setscbbatch``` to be present.

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

**setscainteractive**

```
channel  baud  options  setscainteractive  –
```

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

**Errors:** rangecheck, stackunderflow, typecheck

### 5.2.4 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 ```ImagingBBox```. 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. The only error that is generated is a **limitcheck** caused by insufficient memory for the requested imaging area. In the table below, points (1/72 inch) are used as the units for the **PageSize** and **ImagingBBox**.

**Table 5.7 Paper size compatibility operators**

<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>lettersmall *</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]</td>
<td>null</td>
</tr>
<tr>
<td>note *</td>
<td>[width height]</td>
<td>[25 25 width-25 height-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**, **lettersmall**, and **lettertray** 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** and **legaltray** compatibility operators 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**, **a4small**, and **a4tray** 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** and **b5tray** compatibility operators 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.
5.2.5 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. Also, a limitcheck error can occur because of insufficient memory for the requested imaging area.

<table>
<thead>
<tr>
<th>Operator</th>
<th>PageSize</th>
<th>ImagingBBox</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>
<tr>
<td>b5tray</td>
<td>[516 729]</td>
<td>null</td>
</tr>
<tr>
<td>11x17tray</td>
<td>[792 1224]</td>
<td>null</td>
</tr>
</tbody>
</table>

5.2.6 Page Duplex Compatibility Operators

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

duplexmode

- duplexmode bool

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**

- **newsheet**

forces a new sheet to be started.

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

**Errors:** none

**setduplexmode**

`bool setduplexmode` —

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` —

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
5.2.7 Device Compatibility Operators

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

**devdismount †**

```plaintext
string devdismount –
```

Sets to false the Mounted device parameter boolean of the parameter set corresponding to the device specified by string. It is necessary for the device to be mounted before it can be dismounted. Trying to dismount a device that is not mounted will have no affect. Some devices cannot be dismounted. Trying to dismount these will also have no affect.

**Errors:** invalidaccess, stackunderflow, undefinedfilename

**devforall †**

```plaintext
proc scratch devforall –
```

devforall enumerates all known storage devices.

For each storage device, devforall copies its name into the supplied scratch string, pushes a string object that is the substring of scratch that was actually used, and calls proc. devforall does not return any results of its own, but proc may do so.

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

**devformat †**

```plaintext
string pages format devformat -
```

Sets the LogicalSize device parameter boolean of the parameter set corresponding to the device specified by string to the value specified by pages. Sets the InitializeAction, in the same parameter set, to the value of format plus one. Refer to the InitializeAction and LogicalSize filesystem device parameters for complete details.

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

`string devmount bool`

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.

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

**devstatus**

`string devstatus false` (if device not found)

`string devstatus searchable writeable hasNames mounted removable searchOrder freePages size true` (if device found)

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`. The attributes are described in the following table.

**Table 5.9 Attributes for compatibility operator devstatus**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchable</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>writeable</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>hasNames</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>mounted</td>
<td>The <code>mounted</code> boolean (<code>Mounted</code> device parameter) indicates whether the device is mounted.</td>
</tr>
<tr>
<td>removable</td>
<td>The <code>removable</code> boolean (<code>Removable</code> device parameter) indicates whether the device can be removed.</td>
</tr>
<tr>
<td>searchOrder</td>
<td>The <code>searchOrder</code> attribute corresponds to the <code>SearchOrder</code> device parameter and indicates the priority at which the device participates when searching for a file in operations in which no device has been specified.</td>
</tr>
</tbody>
</table>
The `freePages` boolean (Free device parameter) indicates the amount of free space (in pages).

The `size` attribute (Size device parameter) indicates the current size of the PostScript file system (in pages).

For a complete description of each of the device parameters mentioned above, refer to Table 3.10.

**Errors:** stackunderflow