List of Figures

Figure 2.1  Default user space orientation for a portrait envelope PageSize  34
Figure 3.1  Relationship between network communications protocols and physical communications medium  52
Figure 3.2  Relationship between the communication parameter sets  55
Figure 3.3  Communications parameters sets using NV values  56
Figure 3.4  Communications parameter sets using “hard wired” values  57
List of Tables

Table 2.1  Page device parameters  6
Table 2.2  PageSize entry in the Policies dictionary  14
Table 2.3  Fax options dictionary entries  20
Table 2.4  Entries in a dictionary passed to custom fax job procedures  28
Table 3.1  User parameters  39
Table 3.2  System parameters  41
Table 3.3  Parameters common to all device parameter sets  51
Table 3.4  Parameters present in parameter sets of type /Communications  58
Table 3.5  Parameters present in serial communication parameter sets  62
Table 3.6  Parameters present in parallel communication parameter sets  66
Table 3.7  Parameters present in SCSI communication parameter sets  68
Table 3.8  Parameters present in LocalTalk communication parameter sets  69
Table 3.9  Parameters present in EtherTalk communication parameter sets  71
Table 3.10  Node Address Forms  73
Table 3.11  Parameters present in the %LPR% communication parameter set  74
Table 3.12  Parameters present in the %AppSocket% communication parameter set  76
Table 3.13  Parameters present in the %Telnet% communication parameter set  79
Table 3.14  Parameters present in the %RemotePrinter% communication parameter set  80
Table 3.15  Parameters present in the %PrintServer% communication parameter set  81
Table 3.16  Parameters present in the %SNMP% parameter set  83
Table 3.17  Parameters present in the %Syslog% parameter set  85
Table 3.18  Parameters present in the %TCP% parameter set  86
Table 3.19  Parameters present in the %UDP% parameter set  87
Table 3.20  Parameters present in the %IP% parameter set  88
Table 3.21  Parameters present in the %SPX% parameter set  92
Table 3.22  Parameters present in the %IPX% parameter set  93
Table 3.23  Parameters present in the %EthernetPhysical% parameter set  95
Table 3.24  Parameters common to device sets of type /FileSystem  97
Table 3.25  Parameters present in disk (/FileSystem) devices  97
Table 3.26  Parameters present in cartridge or rom (/FileSystem) devices  101
Table 3.27  Parameters present in the %os% parameter sets  104
Table 3.28  Parameters present in the %Scsi% parameter sets  106
Table 3.29  Parameters present in the %Engine% parameter set  108
Table 3.30  Parameters present for the LaserJet III emulator  110
Table 3.31  Parameters present for the LaserJetIIP emulator  113
Table 3.32  Parameters present for the color version of the HP7475A plotter emulator  114
Table 3.33 Parameters present for the Diablo630 emulator 115
Table 3.34 Parameters present in the %Fax% parameter set 116
Table 3.35 Parameters present in the %Calendar% parameter set 123
Table 3.36 Operators for accessing fax job logs 124
Table 3.37 Entries in a job dictionary 125
Table 3.38 Entries in a Calls dictionary 126
Table 3.39 Operators for accessing job queues 127
Table 3.40 transmitjobsforall generated dictionary 128
Table 3.41 receivejobsforall generated dictionary 129
Table 3.42 Entries in a translationDict 131
Table 4.1 Regular resources 137
Table 4.2 Resources whose instances are implicit 138
Table 4.3 Resources used in defining new resource categories 140
Table 4.4 Description of keys present in an instance of the category OutputDevice 141
Table 4.5 Possible instances of the HWOptions resource category 142
Table 5.1 Entries in a type 6 halftone dictionary 144
Table 6.1 statusdict compatibility operators 149
Table 6.2 Stop bits 161
Table 6.3 Data bits 161
Table 6.4 Flow control 161
Table 6.5 Parity 161
Table 6.6 Options byte to devparams conversion 162
Table 6.7 devparams to options byte conversion 162
Table 6.8 SCC compatibility operators 163
Table 6.9 Paper size compatibility operators (in userdict) 165
Table 6.10 Paper tray compatibility operators 166
Table 6.11 Page duplex compatibility operators 167
Table 6.12 Device compatibility operators 168
Table 6.13 Imagesetter compatibility operators found in statusdict 171
1.1 Introduction

The purpose of the PostScript™ Language Supplement for Version 2014 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 output devices.

Level 2 of the PostScript language 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 language 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 language 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 Language Level 2 Parameters

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

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

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

1.1.2 PostScript Language Level 2 Resources

In Level 2, PostScript language 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 2014 are listed in Chapter 4.

1.1.3 Other Extensions to PostScript Language Level 2

The Postscript Language continues to evolve. Chapter 5 lists those extensions to the language for version 2014.

1.1.4 Compatibility Operators

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

Throughout this manual, the following terms are used:

*device*: A device is defined as a piece of hardware under the control of a PostScript interpreter. There are several categories of devices:

*page device*: A page device can be, for example, a laser print engine producing paper output.

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

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

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

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

*PostScript language product*: A PostScript language 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 language printing device.

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

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

Two operators, \texttt{currentpagedevice} and \texttt{setpagedevice}, respectively read and set the parameter values.

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

\begin{itemize}
  \item \texttt{AdvanceDistance}
  \item \texttt{AdvanceMedia}
  \item \texttt{BeginPage}
  \item \texttt{Collate}
  \item \texttt{CutMedia}
  \item \texttt{Duplex}
  \item \texttt{EndPage}
  \item \texttt{HWResolution}
  \item \texttt{ImagingBBox}
  \item \texttt{Install}
  \item \texttt{Jog}
  \item \texttt{ManualFeed}
  \item \texttt{MediaColor}
  \item \texttt{MediaType}
  \item \texttt{MediaWeight}
  \item \texttt{MirrorPrint}
  \item \texttt{NegativePrint}
  \item \texttt{NumCopies}
  \item \texttt{Orientation}
  \item \texttt{OutputAttributes}
  \item \texttt{OutputFaceUp}
  \item \texttt{OutputType}
  \item \texttt{PageSize}
  \item \texttt{Policies}
  \item \texttt{Separations}
  \item \texttt{Tumble}
\end{itemize}

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

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bind</td>
<td>integer</td>
<td>This parameter requests that the document be bound. The job will be bound</td>
</tr>
<tr>
<td></td>
<td></td>
<td>at a specific time indicated by an integer code:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Code</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>


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

**Booklet** boolean This parameter requests that the document be stapled, trimmed, and folded into booklet form.

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

**DeferredMediaSelection** boolean This page device key is found on those devices that support 2 different media models. A value of *false* indicates that media selection is to be done as described in section 4.11.4 of the *PostScript Language Reference Manual, Second Edition*. A value of *true* indicates that the product will be completely responsible for verifying the media requests. The motivation for this is usually that the product is handing the requests off to another “entity” that will guarantee the media requests will be satisfied at printing time. This is the reason for the name **DeferredMediaSelection**—the actual selection of the media is deferred to a time later than the time of this execution of setpagedevice.

**Note** When **DeferredMediaSelection** is true and the **PageSize** is rejected by a product, **PageSize** policies 0 and 3 through 6 will result in a configuration error. Also, if **PageSize** Policy 7 is being used and **DeferredMediaSelection** is true, it is product dependent as to whether or not media will be pulled from the current tray.
When `DeferredMediaSelection` transitions between `true` and `false`, some media parameters may need to be initialized again. For example, each media selection model may have its own notion of the default page size that should be used if `PageSize` has not been explicitly specified.

`DeviceRenderingInfo` dictionary
This dictionary provides a location for individual OEMs or products to specify their own device rendering parameters. The only pre-specified key is `Type` of type integer.

`ExitJamRecovery` boolean
If `true`, pages that jam in the exit path are reprinted. If `false` (jam recovery disabled), performance might be improved because more overlapping of page processing is possible.

`FaxOptions` dictionary
For a description of the `FaxOptions` key as well as the fax interface, see “PostScript Language Interface to Fax,” section 2.2.

`Fold` integer
This parameter requests that the document be folded. The job will be folded at a specific time indicated by an integer code:

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


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

`ImageShift` array
This array, `[x y]`, specifies the distance, in default user space, that each image on a page is to be shifted in the x direction for ‘x’ units and in the y direction for ‘y’ units. For page images which are to appear on a front side, the horizontal shift is to the right if x > 0, or to the left if x < 0. The vertical shift is to the top if y > 0, or to the bottom if y < 0. For page images which are to appear on a back side, the horizontal shift is to the left if x > 0, or to the right if x < 0. The vertical shift is to the bottom if y > 0, or the top if y < 0.

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

The InputAttributes dictionary normally contains an entry for each input media source. The entry consists of an integer representing the input paper slot and an associated dictionary. Some products may support the InsertSheet boolean entry in the slot dictionaries that are sub-dictionaries of InputAttributes. This entry indicates whether or not the slot holds special insert sheet media. InsertSheet is used during the media matching process and compared against the setpagedevice key InsertSheet (described next).

**InsertSheet** boolean

This parameter specifies whether or not to select inserted media.

setpagedevice compares it with the InsertSheet value, if any, in the InputAttributes entries for all media that it considers. Refer to section 4.11.4, “Media Selection” in the PostScript Language Reference Manual, Second Edition, keeping in mind that InsertSheet is also an input media entry found within the InputAttributes dictionary.

A side-effect of executing setpagedevice with InsertSheet equal to true and selecting an insert sheet slot is that the imageable area gets set to a zero-area region to ensure that nothing is imaged on the inserted sheet. That is, the insert sheet is explicitly not imaged. The insert sheet slot has the special property that the media coming from this slot is not sent through the fuser on its way to the output bin. The media pulled from an InsertSheet slot does not go through the normal paper path. 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.

Here is an example of how to use InsertSheet.

```plaintext
%... PostScript language code for page n
%... page n+1 is an inserted sheet
%
save
<</InsertSheet true>> setpagedevice % selects InsertSheet media
showpage
%send the InsertSheet media on to the output bin as page n+1
restore %implicitly go back to using the regular media
%
%...PostScript language code for page n+2
```

**ManualFeedTimeout** integer

The number of seconds the printer waits for a page to be fed manually before generating a timeout error. A zero value means no timeout (infinite wait).
Margins array If the device supports multiple resolutions (that is, different values of \texttt{HWResolution}), the margin values are interpreted according to some canonical default resolution and are scaled appropriately at other resolutions. This ensures they represent the same physical distance when the resolution is varied. The canonical default resolution is product dependent and specified in the Addendum for the product. For more information on \textit{Margins}, see Table 4.11 in section 4.11.3 of the PostScript Language Reference Manual, Second Edition.

MediaPosition integer or null This is an integer that indicates the slot that is to be used. The interpretation of the integer is product specific since slot numbers themselves are product specific. If media matching is in effect, \textit{MediaPosition} does not override the matching process but does alter (making use of policy whenever possible) it so that if the requested slot number can at all be chosen in a manner that is consistent with media matching, it will be used. If media matching is not in effect (e.g., \texttt{DeferredMediaSelection} is supported and true) then it is a product decision as to how to resolve potential conflicts between the various media requests and the \textit{MediaPosition} request. If \textit{MediaPosition} is set to null, it plays no role in the media selection process. If the \texttt{PageSize} policy is 7 or if manually feeding, \textit{MediaPosition} is ignored.

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

When the value of \textit{OutputDevice} changes, the usual inheritance of values not specified in the operand to \texttt{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 \texttt{setpagedevice} can change when changing the value of \textit{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 \textit{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 \texttt{showpage} instead of being overlapped with execution of subsequent pages. This facilitates measuring the complete cost of page execution.
PageOffset

array This array, [x y], contains two real numbers which are used to relocate the page image on the media x units in the device x coordinate direction and y units in the y device coordinate direction. x and y are always expressed in units of 1/72 of an inch. This positioning is typically accomplished by altering the current transformation matrix (CTM). However, on some products, this positioning can be accomplished by device-dependent means that are independent of the graphics state (the CTM in particular). The PageOffset key is typically found on imagesetters and is used to control where the image is to appear on the media.

PostRenderingEnhance

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

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

PreRenderingEnhance

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

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

ProcessColorModel

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

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

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

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

**SeparationColorNames**

This parameter specifies those Separation color spaces that the device supports. This array can contain either names or strings, for example, `/Pink /Green` or `[(Pink) (Green)]` or `/Pink (Green)`. If the name used in a `/Separation name ... ` setcolorspace` operation is included in this array, that colorant will be used, rather than the alternate color space. Any other color will be mapped to one or more of the named colors through the alternate color space and tintTransform procedure of the setcolorspace operator. This is described in section 4.8.4 of the PostScript Language Reference Manual, Second Edition.

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

• for **DeviceCMY**, the empty array `[]` is equivalent to `/Cyan /Magenta /Yellow`.

• for **DeviceCMYK**, the empty array `[]` is equivalent to `/Cyan /Magenta /Yellow /Black`.

• for **DeviceRGB**, the empty array `[]` is equivalent to `/Red /Green /Blue`.

• for **DeviceRGBK**, the empty array `[]` is equivalent to `/Red /Green /Blue /Black`.

• for **DeviceGray**, the empty array `[]` is equivalent to `/Gray`.

**SeparationOrder**

If separations are being made, this parameter specifies that they be produced in the order given by the array of color names (where an array can contain either names or strings, such as `/Cyan /Magenta` or `[(Cyan) (Magenta)]`). Legal values are the names of the colorants of the native color space, as well as any additional names specified by **SeparationColorNames**.
A separation will be produced for each occurrence of a name; multiple occurrences will produce multiple separations. No separations will be produced for colors whose names are not given regardless of their appearance in \texttt{SeparationColorNames}. The named separation color space is defined (as opposed to reverting to the alternative color space), but the output for that separation is discarded when a certain color name is not given.

An empty array \([\ ]\) requests that separations for all colors of the native color space, as well as all colors requested by \texttt{SeparationColorNames}, be produced in an unspecified order.

When not making separations, some devices may use \texttt{SeparationOrder} to determine the colorants and the order in which they are to be applied to the composite image.

\begin{itemize}
  \item \textbf{Signature} boolean If \texttt{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 \texttt{Signature} value of \texttt{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 \texttt{Signature} is supported by relatively few products and is subject to resource limits in products that do support it.
  \item \textbf{SlipSheet} integer This parameter requests that slip sheets (slip sheet media selection is product specific) be inserted. There is no way to render a slip sheet; the engine is simply told when to insert them. For example, a slip sheet can be a colored sheet of paper that visually separates multiple copies or a tabbed sheet that separates chapters. Compare with the description of \texttt{InsertSheet} above.
\end{itemize}

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

\begin{center}
\begin{tabular}{|c|p{10cm}|}
\hline
\textbf{Code} & \textbf{Meaning} \\
\hline
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. \\
4 & Insert slip sheet after each \texttt{showpage} or \texttt{copypage}. \\
\hline
\end{tabular}
\end{center}

See the descriptions of \texttt{Collate} and \texttt{Jog} in section 4.11.3 of the \textit{PostScript Language Reference Manual, Second Edition}, for a description of the above terminology.
Staple integer This parameter requests that the job be stapled. The job will be stapled at a specific time indicated by an integer code:

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


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

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

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

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


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

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>PageSize</td>
<td>integer</td>
<td>This parameter specifies the recovery policy to use when the PageSize cannot be matched (within a tolerance of 5 units) with any available media. The policy value 7 has been added to Table 4.14 in section 4.11.5, “Policies,” in the PostScript Language Reference Manual, Second Edition. 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.</td>
</tr>
</tbody>
</table>

2.1 Details Dictionaries

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

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

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

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 are checked to be syntactically correct only if the Type value matches what is expected by the printing device. If the Type key is not specified in the Details dictionary, an undefined error will result. When the Type value is a number not known by the printing device, policy is consulted. When the Type value is known, the validity of the values within the Details dictionary are only checked if the feature will be enabled for the page device in effect as a result of setpagedevice. As with all page device entries, syntactically incorrect settings result in appropriate PostScript language errors (for example, typecheck) and invalid values result in policy being consulted.

The Type Entry

Every Details dictionary has a Type entry whose integer value completely determines how the Details dictionary entries 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. The criteria for merging versus overwriting is product dependent. Details dictionaries and their associated Type entries are registered by Adobe Systems.
2.2 PostScript Language Interface to Fax

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

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

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

- “PostScript Language Interface to Fax,” section 2.2, describes the interface which controls individual fax jobs. “Some Sample Fax Jobs,” section 2.2.7 presents several example raster and PostScript fax jobs.

- “The Fax Environment Interface,” section 3.6, describes the device parameters and administrative resources needed to set, control, and examine the fax environment shared by all jobs and all users connected to the fax printer.

2.2.1 The Fax Job Interface

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

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

Fax Job Transmissions

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

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

It is possible to request that a transmission take place at a particular time. These delayed transmissions, of course, are made with a single call. Since the pages of a raster job are prepared when the job is submitted, all of the pages of a delayed raster transmission must fit in storage otherwise the job will fail.

The transmission of a PostScript language file to a single destination is always made in a single call. This call takes place only after the entire file is in storage. If the file will not all fit in storage, the job will fail.

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

### 2.2.2 Sending Raster Images

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

\begin{verbatim}
  <</OutputDevice /Fax other key-value pairs >> setpagedevice
\end{verbatim}

With a call to \texttt{setpagedevice}, one of three cases is possible:

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

2. A new page device is established when there is no current page device. In this case, entries in the new page device which are not specified in the argument dictionary to \texttt{setpagedevice} are initialized to product-dependent default values.
3. A new page device is established that changes the value of OutputDevice in the current page device. In this case, entries in the new page device are not inherited from the current page device, rather the entries in the new page device which are not specified in the argument dictionary to setpagedevice are initialized to product dependent default values.

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

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

The FaxOptions DialCallee Key

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

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

2.2.3 Transmitting PostScript Language Files

It is possible to transmit jobs as PostScript language files between consenting PostScript fax printers. Two entries in the argument dictionary passed to the faxsendps operator control this transmission. These are RevertToRaster and PostScriptPassword. The %Fax% device’s ReceivePostScript and PostScriptPassword parameters (see Table 3.34 on page 116) control the receipt of PostScript language jobs by a fax printer.
A job may be sent in PostScript language form by using the `faxsendps` operator found in the `FaxOps ProcSet` instance:

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

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

1. The `file` is read to EOF with the contents of the file being saved internally. If there is sufficient storage to hold the entire file, proceed to step 2, below.
   
   If there is not sufficient storage to hold the entire file, `RevertToRaster` is consulted. If `RevertToRaster` is `false`, the job fails and the message “Storage to assemble PostScript transmission exhausted” will be sent to `stdout`. If it is `true`, then the local machine will execute the equivalent of

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

   resulting eventually in a raster image fax transmission. The process terminates here, and steps 2 through 4 do not occur.

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

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

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

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

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

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

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

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

#### Table 2.3 Fax options dictionary entries

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CalleePhone</strong></td>
<td>string or null</td>
<td>This key indicates the human-readable telephone number of the callee fax machine. The value of <strong>CalleePhone</strong> is used for the <strong>Confirmation</strong>, <strong>CoverSheet</strong>, and <strong>PageCaption</strong> procedures. It differs from <strong>DialCallee</strong> primarily in that it omits or alters routing prefixes and suffixes. For example, the Swiss number in the example under <strong>DialCallee</strong> might be written here (0041-5-55-55-55732) or (CH 5-55-55-55732). If the value of <strong>CalleePhone</strong> is null, the value of <strong>DialCallee</strong> is used.</td>
</tr>
<tr>
<td><strong>CallerID</strong></td>
<td>string or null</td>
<td>The ID is defined by the CCITT (1988) fax protocol and is a string of up to 20 characters which the caller uses to identify himself to the callee. If it has a null value (the default), then the value of <strong>ID</strong> from the %Fax% device parameter set will be used (see Table 3.34 on page 116). If <strong>ID</strong> is not set, then the string returned by the system parameter <strong>PrinterName</strong> will be used. If this string is greater than 20 characters, then the left-most 20 characters will be used.</td>
</tr>
<tr>
<td><strong>CallerPhone</strong></td>
<td>string or null</td>
<td>This key indicates the human-readable telephone number of the caller fax machine.</td>
</tr>
</tbody>
</table>
**Confirmation**

procedure or null

Prints a confirmation sheet (on the local sending print mechanism) for this job. The **Confirmation** procedure is executed when the fax job has finished and the transmission, if any, has completed. You can omit confirmation reports (by setting **Confirmation** to null) or customize them (by providing your own procedure). You may, for example, want to print a report only if there is an unusual printer status.

The **Confirmation** procedure must explicitly call the **showpage** operator to print the page. This allows confirmation to be sent to the host computer instead of being printed. Therefore, if an alternate definition of the confirmation report is to print upon job completion, the procedure definition must include a call to **showpage**. The details of writing these procedures are given in section 2.2.5, “CoverSheet, Confirmation, and PageCaption Procedures”.

The default value depends on the value of the %Fax% parameter **DefaultConfirmOn**. If **DefaultConfirmOn** is true (see Table 3.34 on page 116), then the default value is

```
{/DefaultConfirmation faxopsexec}
```

which provides a simple, generic report. If **DefaultConfirmOn** is false, then the default value of **Confirmation** is null.

**Copies**

array of dictionaries or null

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

**CoverNote**

array of strings or null

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

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

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

The default value for this key depends on the %Fax% parameter, DefaultCoversOn, (see Table 3.34 on page 116). If this parameter is true, then the default is

`{/DefaultCoverSheet faxopsexec}`

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

The CoverSheet procedure should not include a showpage operator. showpage is called automatically after the CoverSheet procedure is invoked.

**CoverSheetOnly**

boolean  This key indicates that it is all right to send an “empty” job. If this boolean is false and the PostScript language job produces no pages, then no phone call is made and nothing is sent. If this boolean is true and CoverSheet is non-null, then the page generated by the cover sheet procedure is sent in any case. The default value is false.

**DialCallee**

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>begin pulse dialing (initial default)</td>
</tr>
<tr>
<td>T</td>
<td>begin DTMF (Touch-Tone) dialing</td>
</tr>
<tr>
<td>0-9</td>
<td>send signal digit to telephone exchange</td>
</tr>
<tr>
<td>* #</td>
<td>send DTMF symbol to telephone exchange</td>
</tr>
</tbody>
</table>
Note: A comma internal to a dialing string will cause dialing to pause for 2 seconds before proceeding. A comma at the end of the string will cause the system to pause for up to 10 seconds (more) waiting for the call to make its way through all switches involved and a connection to be made. This can be necessary, for example, when making international calls in which the time elapsed before ringing starts is considerably longer than with domestic calls. In certain countries, limits may automatically be placed on how much of a delay is actually possible at the end of dialing. Thus the final commas should be viewed as requests rather than absolute commands.

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

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

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

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

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

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

If the FaxType is 0, then the transmitted y-resolution will be approximately 100 lines per inch; if 1, the y-resolution will be twice as fine (and the transmission time will be correspondingly longer). As the CCITT standards evolve, other integers may trigger other modes. If the value associated with FaxType is null, then the value of the %Fax% device parameter DefaultResolution, which is 0 or 1, will be used to select the resolution (see Table 3.34).

**HostJobID** integer This key allows jobs to be tagged with identifiers that can then be used to track the progress of those jobs. The value of the key is saved in the transmission logs and it is also carried with a job as it progresses through the
various stages of processing and transmission. The value of this key is returned in the dictionary presented by the `transmitjobsforall` (see page 124) operator from the `FaxAdminOps ProcSet` instance. The default value is 0.

**MailingTime**
array of integers or null

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

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

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

**MaxRetries**
integer or null

This key indicates how many additional tries after the first should be made before giving up on the transmission of a fax message. Attempts can fail, for example, because of a busy or no-answer when the call is placed. If the value is null, then the value used is given by the `%Fax%` parameter `DefaultRetryCount` (see Table 3.34). The maximum value is 100. If an attempt is being made to send a PostScript language file and the job reverts to raster, the count of the number of failed calls, in effect, is reset to 0. This is because reverting is equivalent to submitting a new job.

**nPages** integer or null

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

**PageCaption**
procedure or null

This PostScript language procedure generates information lines on the top of transmitted fax pages. Typically, such information will contain the sender’s name, the recipient’s name, and the current page number. The `PageCaption` procedure should not call the `showpage` operator since the procedure only makes alterations to the existing page image. The generation of captions will
not occur if \texttt{PageCaption} is \texttt{null}. For more information on the interface to a \texttt{PageCaption} procedure, see section 2.2.5, “CoverSheet, Confirmation, and PageCaption Procedures.”

The default value for this key depends on the \texttt{%Fax%} parameter, \texttt{DefaultCaptionOn} (see Table 3.34). If this parameter is \texttt{true}, then the default is the procedure

\texttt{/DefaultPageCaption faxopsexec}

If \texttt{DefaultCaptionOn} is \texttt{false}, then the default value is \texttt{null}.

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

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

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

\textbf{RecipientLanguage} \texttt{string or null} \ The value is the name of the natural language to use when preparing cover sheets and page captions. If the value is \texttt{null} or a translation dictionary for the named language cannot be found, the value of the \texttt{%Fax%} device parameter \texttt{LocalLanguage} will be consulted. If there is a translation dictionary for the named language it will be used, otherwise the one for English will be used. See section 3.6.5 on page 129 for more information on translation dictionaries.

\textbf{RecipientMailStop} \texttt{string or null} \ This key contains information helpful for hand delivery of the fax message. For example, Mail Stop 23A, Bldg. 19.

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

**RecipientOrg**

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

**RecipientPhone**

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

**Regarding**

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

**RetryInterval**

integer or null  
This key is a positive integer that specifies the number of minutes to wait
before retrying to send a fax that failed. If the value is null, then the value
is determined by the %Fax% device parameter **DefaultRetryInterval** (see
Table 3.34); the maximum value is 60 minutes.

**RevertToRaster**

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

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

**SenderMailStop**

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

**SenderName**

string or null  
This key indicates the document’s sender; for example, Dr. Jane Green. A
null value for **SenderName** causes the software to seek an alternative non-
null value to store in the job log. The first alternative is to use the value of
**SenderOrg**. If **SenderOrg** has a null value then **CallerID** is used. If this too
is null, then the value of the %Fax% device parameter **ID** is used.
SenderOrg
string or null
This key indicates the sender’s company or organization name. If SenderOrg is null, then the value of SenderName is used. If SenderName is null, the value of the %Fax% device parameter ID is used (see Table 3.34).

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

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

2.2.5 CoverSheet, Confirmation, and PageCaption Procedures

The CoverSheet, Confirmation, and PageCaption procedures in the options dictionary can be used to customize individual fax jobs. If these are not given, default procedures will be used. The default procedures are contained in a writable ProcSet instance named FaxDefaultProcs. By redefining entries in this ProcSet instance outside the server loop, it is possible to change the default report procedures. Initial VM is built with this ProcSet instance pointing at the product’s built-in procedures.

The report procedure entries contained in FaxDefaultProcs are

<table>
<thead>
<tr>
<th>Key</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultCoverSheet</td>
<td>{/InternalDefaultCoverSheet faxopsexec}</td>
</tr>
<tr>
<td>DefaultPageCaption</td>
<td>{/InternalDefaultPageCaption faxopsexec}</td>
</tr>
<tr>
<td>DefaultConfirmation</td>
<td>{/InternalDefaultConfirmation faxopsexec}</td>
</tr>
<tr>
<td>DefaultReportJobList</td>
<td>{/InternalReportJobList faxopsexec}</td>
</tr>
</tbody>
</table>

The last item here, DefaultReportJobList, is the procedure behind the reportjoblist FaxAdminOps operator. Activity reports are produced using this operator.

For more information on the FaxDefaultProcs ProcSet instance, see section 3.6.5.

The CoverSheet, Confirmation, and PageCaption procedures should make no change to the global state that would affect the subsequent printed appearance of the PostScript language job. These procedures are called with no parameters. However, there will be two dictionaries on the dictionary stack which provide the information required for these procedures to do their work.
The first dictionary is the `FaxOptions` dictionary. Since the `Confirmation` procedure may be run long after the fax job was processed and since the `OutputDevice` will have been set to `Printer` typically before this procedure is run, it is actually a copy of the `FaxOptions` dictionary from when the fax job was processed that is present on the stack. This dictionary has the `PageCaption` and `CoverSheet` entries removed.

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

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CalleeID</code></td>
<td>string</td>
<td>This string is defined by the CCITT (1988) fax protocol as a string of up to 20 characters with which the callee can identify himself to the caller. It is transmitted between the two stations when they first handshake with each other. It is not present in the dictionaries provided to the <code>CoverSheet</code> and <code>PageCaption</code> procedures. For multiple call transmissions, the value determined on the first phone call is used.</td>
</tr>
<tr>
<td><code>CallLength</code></td>
<td>integer</td>
<td>This key provides the number of seconds that the transmission session lasted. It is not present in the dictionaries provided to the <code>CoverSheet</code> and <code>PageCaption</code> procedures.</td>
</tr>
<tr>
<td><code>CoverType</code></td>
<td>integer</td>
<td>This key is only meaningful to the <code>CoverSheet</code> procedures. It has a value indicating the type of cover needed.</td>
</tr>
<tr>
<td><code>Value</code></td>
<td><code>0</code></td>
<td>front cover</td>
</tr>
<tr>
<td><code>Value</code></td>
<td><code>1</code></td>
<td>back cover</td>
</tr>
<tr>
<td><code>CurrentPageNo</code></td>
<td>integer</td>
<td>This is the number of the page currently being prepared. This value can be used for generating page captions. Cover sheets are not included in this running count.</td>
</tr>
<tr>
<td><code>ErrorArray</code></td>
<td>array of strings</td>
<td>This array describes particular error conditions. It is indexed by <code>ErrorIndex</code>.</td>
</tr>
<tr>
<td><code>ErrorIndex</code></td>
<td>integer</td>
<td>This integer can be used to retrieve a string describing a particular error condition from the <code>ErrorArray</code>.</td>
</tr>
<tr>
<td><code>IncludesFinalPage</code></td>
<td>boolean</td>
<td>This key indicates whether this session is the last and final transmission session for this job. If <code>true</code>, then this session includes the last page of the overall job, if <code>false</code>, it does not. For more information on transmission sessions, see <code>InitialPage</code> and <code>LimitPage</code>.</td>
</tr>
</tbody>
</table>
**InitialPage** and **LimitPage**  
**integer**  
This key determines which pages of an overall job have been or will be sent in this transmission. The job may get broken into more than one distinct transmission session due to errors on the telephone line, overruns or under-runs, and so on. If cover pages are being generated, then each distinct transmission will have its own cover sheet. The two items here can be used by the cover page procedure to figure out how many pages are in a particular transmission and where they fall with respect to other transmissions (if any). **InitialPage** is the page number of the first page of the session less one. (That is, it is 0 if this transmission includes the first page of the job.) **LimitPage** is the total number of pages in this session and all preceding sessions of this job excluding all cover sheets.

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

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

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

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

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

For confirmation reports, this is the time of the first call (of possibly many) made for the job. For cover sheets and page captions, this is the time the PostScript language job was submitted and processed.

---

### 2.2.6 Broadcast Transmission of Faxes

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

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

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

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

If a raster is broadcast, individual cover sheets will be produced for each recipient, as determined by the `CoverSheet` procedures in the merged `FaxOptions` dictionaries. These cover sheets are imaged separately from the common pages and may contain recipient-specific information derived from the merged `FaxOptions` dictionaries.

When a PostScript language file is being broadcast, the first part of the common file sent to all recipients contains code to reproduce a dictionary. This dictionary is used by the receivers to create a cover page. The dictionary that is coded and sent is the argument dictionary passed to the `faxsendps` operator, not one of the merged dictionaries.

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

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

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

2.2.7 Some Sample Fax Jobs

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

A Simple Example Sending a Raster File

This snippet of PostScript language code produces a cover page (if the value of the %Fax% parameter DefaultCoversOn is true; see Table 3.34) and the document page on the fax machine answering the phone at 415-555-3710.

```
2 dict dup
begin
 /OutputDevice /Fax def
 /FaxOptions 10 dict dup begin
   /DialCallee (T9,1-415-555-3710) def
   /RecipientName (Joe Smith) def
   /FaxType 0 def
   def
 end
def
end
setpagedevice

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

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

An Example with User-defined Procedures

This is an extension of the previous example, and shows the use of custom cover sheet, page caption, and confirmation procedures.
This example sends the same fax message as the first example but does it by sending the PostScript language file instead of a rasterized image.
currentfile
10 dict dup begin
   /DialCallee (T9,1-415-555-3710) def
   /RecipientName (Joe Smith) def
   /FaxType 0 def
   /RevertToRaster true def
end

/FaxOps /ProcSet findresource /faxsendps get exec

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

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

2.3 Envelope Orientation in User Space

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

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

Figure 2.1 illustrates two envelopes: one with its flap along the long edge of the envelope, and one with its flap along the short edge of the envelope. The dashed line indicates that the flap is on the side of the envelope facing down.

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

If the flap is along the short edge of the envelope, then the default user space for a portrait PageSize is set up as in panel B of Figure 2.1.
For landscape PageSize values ([width height] with width > height), the orientation of default user space is defined relative to the orientation for portrait PageSize values. This relationship is described in Table 4.10 in section 4.11 of the PostScript Language Reference Manual, Second Edition.

2.4 Errors Generated by Page Device Parameters

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

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

2.4.1 typecheck Errors

A typecheck error is generated if:

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

  ```
  << /BeginPage 4 >> setpagedevice
  This example generates a typecheck error.
  ```
• A literal array is given for a value that should be a procedure. However, an executable array is acceptable wherever an array value is expected. Packed arrays are always acceptable wherever an array is acceptable. The first two examples below would generate a typecheck error, and the third would not:

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

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

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

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

true setpagedevice
This example generates a typecheck error.

2.4.2 rangecheck Errors

A rangecheck error is generated if:

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

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

<< /InputAttributes << 0 << /PageSize [600 700 800] >> >> >> setpagedevice
This example generates a rangecheck error.

• A value of the right type, but beyond the acceptable range of values, is given either as the value for a feature, or as a component of a value within a compound value. Each of the following examples would generate a rangecheck error:

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

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

An undefined error is generated if:

- The Type key is not specified in a Details dictionary. For example:

  \[
  \langle /Fold 4 /FoldDetails \langle/FoldType (ZFold) \rangle \rangle \text{ setpagedevice}
  \]
  This example generates an undefined error. The Type key is mandatory in the FoldDetails dictionary and it is missing in this example.

2.4.4 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:

  \[
  \langle /MediaColor (blue) noaccess \rangle \text{ setpagedevice}
  \]
  This example generates an invalidaccess error.

  \[
  \langle /PageSize {612 792} executeonly \rangle \text{ setpagedevice}
  \]
  This example generates an invalidaccess error.

  \[
  \langle /BeginPage {pop} executeonly \rangle \text{ setpagedevice}
  \]
  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:

  \[
  \langle /PageSize [612 792] \rangle \text{ noaccess setpagedevice}
  \]
  This example generates an invalidaccess error.
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 operator to read the parameter values and an operator to set the parameter values. The resulting six operators are `currentuserparams`, `setuserparams`, `currentsystemparams`, `setsystemparams`, `currentdevparams`, and `setdevparams`.

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

### 3.1 Two Kinds of Unencapsulated Jobs

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

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

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

- The device parameters `Interpreter` and `Protocol` found in device sets of type `/Communications` do not require a password if one or both are the only entries in the dictionary passed to `setdevparams`. If the only additional key in the dictionary is the password, it is ignored (see `Interpreter` described in Table 3.4 and `Protocol` described in Table 3.5).

3.3 User Parameters

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

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

User parameters are maintained on a per context basis in environments that support multiple contexts.
The following user parameters are described in Table C.1, Appendix C of the *PostScript Language Reference Manual, Second Edition*. The description of these parameters is unchanged.

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccurateScreens</td>
<td>boolean</td>
<td>This parameter controls whether the accurate screen algorithm is used during subsequent executions of the <code>setscreen</code> and <code>setcolorscreen</code> operators. This parameter has no affect on screens established by <code>sethalftone</code>. See section 6.4.4 of the <em>PostScript Language Reference Manual, Second Edition</em>, for a description of accurate screening for the <code>sethalftone</code> operator.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
<tr>
<td>JobName‡</td>
<td>string</td>
<td>This parameter establishes <code>string</code> 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 <code>job</code> field that reports the text of this string. The characters should be within the ASCII printable range, because this information is transmitted across arbitrary communications channels and is intended for display to users.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> Any sequence of byte values up to an implementation-dependent maximum length. However, it should not contain the characters ‘;’ or ‘\’ because they would disrupt the syntax of status messages. If the maximum length is exceeded, the string is truncated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> <code>limitcheck</code>, <code>typecheck</code></td>
</tr>
</tbody>
</table>
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 This parameter indicates the current wait timeout which is the number of seconds the interpreter waits to receive additional characters from the host before it aborts the current job by executing a PostScript language timeout error. A value of 0 indicates an infinite timeout. This parameter is initialized to the value of the **WaitTimeout** system parameter at the beginning of each job.

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

**Errors:** typecheck

3.4 System Parameters

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

System parameters are global to the PostScript language 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 the following lists, as well as in Table 3.2, this symbol is used: \(\dagger\) = this key is typically present in all job server (that is, printer) implementations.

\[\begin{array}{lll}
\text{ByteOrder} & \text{CurDisplayList} & \text{CurFontCache} \\
\text{CurFormCache} & \text{CurOutlineCache} & \text{CurPatternCache} \\
\text{CurScreenStorage} & \text{CurUPathCache} & \text{MaxDisplayList} \\
\text{MaxFontCache} & \text{MaxFormCache} & \text{MaxOutlineCache} \\
\text{MaxPatternCache} & \text{MaxScreenStorage} & \text{MaxUPathCache} \\
\text{RealFormat} & \\
\end{array}\]

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

### 3.4.1 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><strong>BuildTime</strong></td>
<td>integer</td>
<td><em>(Read-only)</em> A time stamp identifying a specific build of the PostScript interpreter. The values returned by <strong>BuildTime</strong> on two different products need not be comparable, and in general, <strong>BuildTime</strong> should only be interpreted in conjunction with the manufacturer’s product documentation.</td>
</tr>
<tr>
<td><strong>Legal values:</strong></td>
<td>Any integer.</td>
<td></td>
</tr>
<tr>
<td><strong>Errors:</strong></td>
<td>none</td>
<td></td>
</tr>
<tr>
<td><strong>CurBufferType</strong></td>
<td>name</td>
<td><em>(Read-only)</em> This parameter (typically found on imagesetters) indicates information about how the raster memory is used. The choices are /Band and /Hybrid. /Band indicates that the system will render to bands (pieces of the page) regardless of the amount of RAM available. When /Hybrid is returned, if <strong>MaxRasterMemory</strong> is large enough so that an entire frame buffer fits for the page, the interpreter will build the full page in one “band”. Otherwise, it will do what it does when this parameter returns /Band.</td>
</tr>
<tr>
<td><strong>Legal values:</strong></td>
<td>/Band, /Hybrid</td>
<td></td>
</tr>
<tr>
<td><strong>Errors:</strong></td>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>
CurInputDevice
string  *(Read-only)* This parameter indicates the name of the communications device corresponding to the current input file for the currently executing PostScript language program. The string that is returned corresponds to the communications device parameter set name whose values are normally stored in RAM: for example, (%Serial%). For more information on communications devices, see section 3.5.2, “Device Parameters Associated with Communications.”

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

**Errors:** none

CurOutputDevice
string  *(Read-only)* This parameter indicates the name of the communications device corresponding to the current output file for the currently executing PostScript language program. The string that is returned corresponds to the communications device parameter set whose values are normally stored in RAM: for example, (%Serial%). For more information on communications devices, see section 3.5.2, “Device Parameters Associated with Communications.”

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

**Errors:** none

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

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

**Errors:** none

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

**Legal values:** Any positive integer

**Errors:** none
CurStoredScreenCache
integer
(Read-only) This parameter indicates the number of bytes currently used for screen files on the storage device which includes the currently active screens.

Legal values: Any positive integer

Errors: none

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

Legal values: true, false

Errors: typecheck

FactoryDefaults boolean
This parameter is usually false. Setting it to true and immediately turning off the printer causes all non-volatile parameters to revert to factory default values at the next power-on. The job that sets FactoryDefaults to true must be the last job executed before power-off; otherwise, the request is ignored. This required physical interaction reduces the chance of malicious jobs resetting the device to factory defaults.

A password is not required in the dictionary passed to setsystemparams if FactoryDefaults is the only entry in the dictionary. This allows the factory defaults to be reestablished even though the system parameters password might have become corrupted. Note that the passwords are among those reset by this operation.

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

Legal values: true, false

Errors: typecheck

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

**Legal values:** Any integer.

**Errors:** none

### FontResourceDir

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

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

Note that applications and users should access external fonts only through the resource operators or **findfont** or, if necessary to access them as files, through **ResourceFileName**. (See the PostScript Language Reference 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

These parameters control the location of external resources for the **Generic** category and all categories based upon it (currently **Category,** **Encoding,** **Form,** **Pattern,** **ProcSet,** **ColorSpace,** **Halftone,** and **ColorRendering**). The **Generic** category implementation concatenates the **Generic-ResourceDir**, 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 This parameter indicates the value in seconds to which the user parameter **JobTimeout** is initialized at the beginning of each job. Trying to set the system parameter **JobTimeout** to a negative value is ignored and the previous setting of **JobTimeout** is used. A value of 0 indicates that the timeout is infinite. Trying to set a number between 1 and 14 will result in 15 being set (i.e., 15 is the minimum value). The reason for the minimum value of 15 is that if small values were allowed, this might prevent a subsequent job from setting **JobTimeout** to another value successfully.

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

**Errors:** 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

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

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

\textbf{Errors:} none

\textbf{MaxImageBuffer}‡ integer This parameter indicates the maximum number of bytes that can be utilized for a single image buffer. An image buffer holds an internal data representation for sampled image source data. The parameter may be rounded by the interpreter if a requested value is out of range.

\textbf{Legal values:} Any integer.

\textbf{Errors:} typecheck

\textbf{MaxRasterMemory} integer This parameter indicates the largest amount of memory, in bytes, that may be allocated to the frame buffer. This parameter may be used to limit the amount of raster memory; unused raster memory is available for use as VM. Thus, \texttt{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). \texttt{MaxRasterMemory} is consulted only during system initialization; any changes to the value of the parameter will not take effect until then.

\textbf{Legal values:} Product dependent.

\textbf{Errors:} typecheck

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

\textbf{Legal values:} Any integer.

\textbf{Errors:} typecheck

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

**Legal values:** -1, 0, or any positive integer

**Errors:** typecheck

MaxStoredScreenCache

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

**Legal values:** 0 or any positive or negative integer

**Errors:** typecheck

MinBandBuffers

integer  This parameter (typically found on imagesetters) is used to specify the minimum number of band buffers (buffers for pieces of a page) to be allocated from memory set aside as raster memory. The default value depends on the product and the amount of memory installed. Typically on imagesetters, the default is 2 for configurations with 32 Megabytes of memory or less and 3 otherwise.

**Legal values:** Any positive integer.

**Errors:** none

PageCount

integer  *(Read-only)* The PageCount parameter indicates the number of pages that have successfully been processed since manufacture. The PageCount parameter is incremented when the interpreter finishes executing each page. The page count is incremented at these times by the value of the current copy count. If one or more pages are not actually printed for any reason, including manual feed timeout and job abort, PageCount is not decreased accordingly. In most products, PageCount is not reset at a user request to return to factory defaults. However, PageCount may be reset if the non-volatile memory in which it is stored has been corrupted.
Note  In releases prior to PostScript Language version 2014, the **PageCount** parameter is incremented when a page completes printing, rather than during execution of the **showpage** or **copypage** operators.

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

**Errors:** none

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

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

**Errors:**   limitcheck, typecheck

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

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

**Errors:** none

**Revision**  integer  *(Read-only)*  This parameter designates the current revision level of the product in which the PostScript interpreter is running. Each product has its own numbering system for revisions, independent of those of any other product. The value is identical to the value of the integer **revision** in **systemdict**.

**Legal values:** Any integer.

**Errors:** none
**StartJobPassword** ‡

**string**

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

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

**Errors:** `limitcheck`, `typecheck`

**StartupMode**

**integer**

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

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

**Errors:** `typecheck`

**SystemParamsPassword** ‡

**string**

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

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

**Errors:** `limitcheck`, `typecheck`

**ValidNV**

**boolean** *(Read-only)*

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

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

**Errors:** `none`
**WaitTimeout** † integer

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

**Legal values:** 0 or any positive integer.

**Errors:** typecheck

---

### 3.5 Device Parameters

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

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

**Note** Not all of the device parameters listed in tables 3.4 to 3.35 will be present in every printer product. Refer to the product addendum for a complete list of parameters supported by any given product.

### Device Parameter Dependencies

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

For example, in the serial communications device set there is an Interpreter and a Protocol parameter. The Interpreter parameter determines which page description language is to be used for an incoming job on that channel. The Protocol parameter determines the communications protocol used to send and receive data. Protocol can be set to /Binary, /Normal, /Raw, or /TBCP. The serial channel cannot be configured to have Protocol set to
3.5 Device Parameters

Device Parameters Set Types

Every device parameter set has a key-value pair which indicates its type. The key is Type and the value can be /Communications, /Emulator, /FileSystem, or /Parameters.

Table 3.3 Parameters common to all device parameter sets

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) This parameter designates the general category of parameters in a device parameter set. Every device parameter set shall contain a Type entry.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: /Communications, /Emulator, /FileSystem, /Parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
</tbody>
</table>

3.5.2 Device Parameters Associated with Communications

A raster output device can have various physical communications channels and can speak many different protocols over these channels. Host computers can communicate with these products by way of diverse network topologies and/or by way of direct point-to-point connections.

Communications Possibilities

There are many ways to communicate with raster output devices. The choices of physical hardware for point-to-point communications include serial, unidirectional and/or bidirectional parallel, and SCSI bus. In the realm of network communications hardware, some choices are Ethernet, Token Ring, LocalTalk and PhoneNET. There are network adaptors (e.g., line multiplexers, Centronics to Ethernet adaptors and SCSI to Ethernet adaptors, etc.) that allow the raster output device to connect to networks to which it
does not otherwise have a direct physical connection. However, the network parameters described later in this chapter are relevant only to raster output devices that are true “peers” of the network with a direct connection to it. These products may support any of the popular protocol stacks, i.e., TCP/IP, Appletalk, and Novell’s Netware® (IPX/SPX). Although the design of most of these protocols predate the Open Systems Interconnection (OSI) Reference Model of the International Standards Organization (ISO), we shall use the OSI model in our description of the parameters associated with the functional layers of these protocol stacks.

There are various device parameter sets defined to aid system administrators in setting up and maintaining network printers. These parameter sets correspond to layers within the OSI model. Only the application layer (of OSI) possesses a parameter set of type /Communications. The PostScript interpreter (or emulator) receives its jobs from this layer. The device parameter sets which correspond to the transport, network, data link and physical layers are of type /Parameters. The parameter set associated with an implementation of the network layer must reference a parameter set associated with an instance of the data link/physical layer (i.e. a distinct interface to the network). For example, if there is only one network layer implementation, but it is connected to n network interfaces, then for each network layer parameter set there is a unique data link/physical parameter set named within it. A parameter set for the network layer can be viewed as a distinct binding of the network address and the network interface.

There are various de facto standard communications protocols which raster output devices must speak in order to reside directly on networks. The choices of protocols and the physical hardware medium on which they reside are limited in certain ways. Figure 3.3 illustrates the present relationship between these protocols and physical communications hardware. The arrows in this diagram indicate that the network protocols can be used with the physical medium pointed to in order to deliver and receive messages.

**Figure 3.1 Relationship between network communications protocols and physical communications medium**
Novell’s Netware® communications protocols were originally derived from the Xerox Network System (XNS) protocols. For example, Novell’s Internet- network Packet Exchange (IPX) is virtually identical to Xerox’s network layer protocol called IDP (Internetwork Datagram Protocol). Novell’s transport protocol is called Se quenced Packet Exchange (SPX).¹

AppleTalk is the name Apple Computer, Inc. chose for their networking software which is built into every Macintosh computer. The data link protocol LocalTalk Link Access Protocol (LLAP) is used while communicating over LocalTalk or PhoneNET networks. (ELAP) EtherTalk Link Access Protocol and (TLAP) TokenTalk Link Access Protocol are the data link protocols for communicating over Ethernet and Token Ring respectively. AppleTalk over Ethernet is called EtherTalk and AppleTalk over Token Ring is called TokenTalk.

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

Refer to Computer Networks by Andrew S. Tanenbaum² for an excellent description of Token Ring and Ethernet.

See Inside AppleTalk® for a description of AppleTalk.³

Before describing various communication parameter sets, let us first think more about how parameter sets of type /Communications are manipulated.

**Communications Parameter Sets**

There can be different ways to set up a product’s communications parameters, including front panels, hardware switches, and PostScript 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.

---

A raster output device typically has several hardware ports for communications. For example, a printer might have a parallel port and two serial ports named channel A and channel B. The parallel port is associated with the parameter device set named `%Parallel%`. Serial channel A, which is wired to a 25-pin RS-232A connector, is associated with the parameter device set named `%Serial%`. Serial channel B, which is wired to either an 8-pin or a 9-pin connector, is associated with the parameter device set named `%SerialB%` or with the parameter set named `%LocalTalk%`. In this example, two device sets are associated with the same port.

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:

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

The three parameter sets for a communications channel have the following general characteristics:

- `%CommName_NV%` values usually are stored in non-volatile memory.
- `%CommName%` values usually are stored in RAM and do not persist when the printer is powered off.
- `%CommName_Pending%` is a read-only parameter set whose values are used to configure the communications hardware and software at the beginning of the next job. This parameter set reflects either the current values of some writable parameter set, such as `%CommName%`, or some predetermined values selected via a switch or front panel. How the system computes the values in `%CommName_Pending%` is described below.
- When there are multiple instances of a certain communications parameter set, the naming convention is `%CommName%`, `%CommNameB%`, `%CommNameC%`, and so on.

The name `%CommName_NV%` is only a hint of actual behavior. In products with limited non-volatile memory, only some of the `%CommName_NV%` set parameters may actually be saved to non-volatile memory, while products with sufficient non-volatile memory typically save all writeable `%CommName_NV%` parameters. PostScript language utility programs need not take these differences into account. If their specific intent is to
affect persistent values, they should use \%CommName_NV\%. The implementa-
tion 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 de-
scribed 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 3.2  Relationship between the communication parameter sets**

Figure 3.2 shows the basic relationship of the three parameter sets.
In this figure, values written to \%CommName\% are written through
to \%CommName_Pending\%; 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 values are to persist across restarting and power cycling). Some
  products store to only one of these sets.

- The product may have switches through which it can be directed to use
  either \%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 3.2, the \texttt{CommName} parameter set, which is in RAM and does not persist when the printer is powered off, is used in many cases (but not all) to update the \texttt{CommName\_Pending} set. Thus, on many products (those with a front panel but no switches), the \texttt{CommName} and \texttt{CommName\_Pending} sets always have the same values and appear redundant.

The \texttt{CommName\_NV} set usually stores the parameters in non-volatile storage. In the simple case in Figure 3.2, writing to \texttt{CommName\_NV} writes through to \texttt{CommName}, which in turn writes through to \texttt{CommName\_Pending}.

In general, a spooler or utility almost always should write to \texttt{CommName}. It should write to \texttt{CommName\_NV} only if parameters are to persist when the printer is turned off.

A front panel usually writes to \texttt{CommName\_NV} to change the power-on parameters, although the front panel also can write to \texttt{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: \texttt{CommName\_NV}, \texttt{CommName\_NV2}, \texttt{CommName\_NV3}, and so on. As is the case with a single non-volatile set, these parameter sets obtain their values by being written to by a PostScript language spooler or utility.

![Communications parameters sets using NV values](image)

Figure 3.3 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 \texttt{CommName\_NV2}, which is indicated by the switch setting. When the switch is set to this
position, or when the product is restarted or powered up with the switch in this position, the values in %CommName_NV2% are written through to %CommName% and to %CommName_Pending%. While the setting %CommName_NV2% is active, a PostScript language job can write to any of the non-volatile parameter sets, but only if it wrote to %CommName_NV2% would the values migrate to %CommName% and %CommName_Pending%. Changing the switch to the position corresponding to %CommName_NV3% would cause %CommName_NV3% values to become the active ones in %CommName% and %CommName_Pending% instead.

**Predetermined Parameter Values**

In addition to the switch settings that indicate which non-volatile parameter set should be used, there can also be other switch settings that short-cut this hierarchy of parameter sets and cause a predetermined set of communications parameters to be written directly to %CommName_Pending%. This situation is shown in Figure 3.4.

**Figure 3.4** 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 communication 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 communic-
ation parameters.

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

Changes to parameters of type /Communications take effect after the current
file (containing one or more PostScript language jobs) is fully processed by
the interpreter and prior to reading from the next file. Note the distinction
between “end of job” and “end of file.” There can be multiple jobs in a file.
See section 3.7.7, “Job Execution Environment,” in the PostScript Lan-

Parameters Present in Parameter Sets of Type /Communications

The parameters listed in Table 3.4 are those found in all device parameter
sets of type /Communications.

Table 3.4 Parameters present in parameter sets of type /Communications

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>DelayedOutputClose</td>
<td>boolean</td>
<td>This parameter selects how the output channel is managed after each job finishes executing. Note that the printer does not wait for the pages of the job to finish printing, but instead immediately starts executing the next job. The DelayedOutputClose boolean parameter is set independently for each communications channel. When DelayedOutputClose is true:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• An EOF is not sent until all pages of a job have been printed. On network channels, the connection remains open until the job finishes printing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If a job produces output, and there are preceding jobs that have not finished printing that are using the same output channel, the output will not be sent until those jobs have completed printing, and the EOFs for them have been sent.</td>
</tr>
</tbody>
</table>
• Spontaneous messages, such as printer error messages, are sent to
the channel if it is either the output channel for the job executing, or the
output channel for jobs that have finished executing but have not finished
printing.

When DelayedOutputClose is false:

• An EOF is sent as soon as a job finishes executing in the interpreter. On
network channels, the connection may be closed when the job finishes
executing, even though pages produced by the job might not have
finished printing.

• Output generated by a job can be transmitted without delay, even if there
are previous jobs using the same output channel that have not finished
printing (note, the EOF for those jobs will have already been sent).

• Spontaneous messages, such as printer error messages, are sent to the
channel only if it is the output channel for the job executing, even if it is
the output channel for previous jobs that have not finished printing.

The DelayedOutputClose setting for a job source is controlled by the
parameter sets for the output channel of that job source. So, if the SerialB
channel is used for the output of jobs received on the Parallel port, then the
DelayedOutputClose value in the SerialB parameter sets applies to jobs
received on both the SerialB and Parallel ports.

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

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

Legal values: true, false

Errors: none

Enabled boolean This parameter designates whether data arriving on the communications
channel 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 is scheduled as an executable job. If Enabled
is false, the data will not be scheduled as an executable job, but the channel
can be used directly by a job for reading and writing data. A configuration-error is generated if setting "Enabled" would produce either of the following situations.

- When trying to set "On" to "false" and "Enabled" to "true" within the same parameter set, a configurationerror will result.

- When trying to turn off "Enabled" in one communications device parameter set results in all channels having "Enabled" set to "false", a configurationerror will result.

Legal values: "true", "false"

Errors: configurationerror, typecheck

HasNames boolean (Read-only) This parameter indicates whether the communications channel represented by the parameter set supports named files. HasNames is always "false" in device parameter sets of type /Communications. This is defined only in device parameter sets of the Type /FileSystem or /Communications.

Legal value: "false"

Errors: none

Interpreter name This parameter designates which interpreter or emulator is to be used to interpret the next incoming job arriving on this communications channel. This parameter is used only if "Enabled" is "true". For certain communication channels there is a relationship between the Interpreter and the Protocol parameters that can result in a configurationerror. See Protocol in Table 3.5 and Table 3.6 for further details.

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

The Interpreter value /AutoSelect is described below. For information on the other legal values, see section 3.5.7, “Emulator Parameters.”

/AutoSelect: The AutoSelect facility provides automatic and seamless switching between the available interpreters and emulators based on the input data stream. The Interpreter parameter should be set to /AutoSelect on channels that connect to hosts which alternately send PostScript language jobs, raw PCL® (LaserJetIIP or LaserJetIII), and “printscreen” jobs (in the IBM PC compatible environment). It can be used on any communications channel. When using AutoSelect for a given communications channel, it is important that the underlying communications protocol is one that preserves all incoming data. In particular, for a serial or parallel channel, this implies that Protocol is set to Raw, Binary, or TBCP.
For serial and parallel communication channels, the following is true.

- **AutoSelect** detects interpreter boundaries and job boundaries if the value of **Protocol** is set to TBCP or Binary.

- **AutoSelect** detects interpreter boundaries, job boundaries, and protocol boundaries and automatically selects the protocol if the value of **Protocol** is set to Raw. This is the recommended setting for **Protocol** when using **AutoSelect**. When **AutoSelect** detects that a PostScript language job is being received and the **Protocol** is Raw, only the Normal and TBCP protocols can be recognized (e.g., Binary is not supported).

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

For other communication channels that are binary in nature, the following is true.

- **AutoSelect** detects interpreter boundaries and job boundaries.

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

**Errors:** configurationerror, rangecheck, typecheck

**On** boolean This parameter designates whether the communication channel is turned on and able to receive and send data. If the parameter is *true*, data transmitted to the channel by a host is buffered and flow control protocols are applied. Data sent to the channel 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 communication devices share the same physical port, and setting the **On** parameter produces a situation in which both channels 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 channel is not considered as a source of jobs to be scheduled, but the channel can be used by a PostScript language job to send and receive data by means of the file operators.

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

**Legal values:**  \(true, false\)

**Errors:**  configurationerror, typecheck

---

### Serial Communication Parameters

Table 3.5 lists those parameters typically found in the device parameter sets named \%Serial\%, \%SerialB\%, \%SerialC\%, and so on.

**Table 3.5 Parameters present in serial communication parameter sets**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baud</strong></td>
<td>integer</td>
<td>This parameter designates the baud rate on the underlying serial hardware. Normally this parameter can be set to any non-negative number; it will not be rounded. The underlying serial hardware will, however, round the baud rate to the nearest achievable value. Hardware rounding will not be reflected in the value of the parameter when it is read. On some products this parameter might be restricted to a small number of legal values.</td>
</tr>
<tr>
<td><strong>CheckParity</strong></td>
<td>boolean</td>
<td>This parameter designates whether parity checking is done by the device on incoming data. This parameter is ignored if the value of <strong>Parity</strong> is None. If <strong>CheckParity</strong> is <strong>true</strong> and a parity error occurs, a PostScript language ioerror results. If <strong>CheckParity</strong> is <strong>false</strong>, no parity checking occurs.</td>
</tr>
<tr>
<td><strong>DelayedOutputClose</strong></td>
<td>boolean</td>
<td>For the general definition of <strong>DelayedOutputClose</strong>, see Table 3.4.</td>
</tr>
</tbody>
</table>

**Legal values:**  \(true, false\)

**Errors:**  rangecheck, typecheck

---

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

**Legal values:**  \(true, false\)

**Errors:**  typecheck

---

**DelayedOutputClose** boolean  For the general definition of **DelayedOutputClose**, see Table 3.4.

**Legal values:**  \(true, false\)

**Errors:**  none
DataBits  integer  This parameter designates the number of data bits per byte communicated over the channel. If this parameter is 7, the high bit of a received byte of data is set to 0. The total number of bits for each byte transmitted or received is the sum of the number of start bits (always 1), data bits, parity bits, and stop bits.

Legal values:  7, 8

Errors:  rangecheck, typecheck

Enabled  boolean  For the general definition of Enabled, see Table 3.4.

Legal values:  true, false

Errors:  configurationerror, typecheck

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

Note  Not all serial channels support all flow control modes.

Following are descriptions of the legal values of FlowControl:

/Dtr: DTR and DSR hardware signals are used by the printing device and the host, respectively to indicate to the other when data may be transmitted. A high value for the signal indicates that data may be transmitted, a low value indicates that data should not be transmitted.

/DtrLow: This parameter is the same as /Dtr except the active sense of the signals is reversed. A low signal indicates that data may be transmitted and a high signal indicates that data should not be transmitted.

/EtxAck: Two characters, ETX and ACK, are reserved for flow control usage. The protocol is symmetric for printing device and host. Each sender knows an agreed upon maximum number of characters that the other side can receive. A sender may send up to this number of characters followed by an ETX. The sender may send more data only when it has received an ACK from the receiver on the other side.

/RobustXonXoff: This protocol operates similarly to the /XonXoff protocol except that periodically (typically every second) the interpreter will send the host an Xon if it is able to receive data.

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

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

**Legal values:** /Dtr, /DtrLow, /ETxAck, /RobustXonXoff, /XonXoff, /XonXoff2

**Errors:** rangecheck, typecheck

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

**Legal value:** false

**Errors:** none

**Interpreter** name For the general definition of Interpreter, see Table 3.4.

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

**Errors:** configurationerror, rangecheck, typecheck

**On** boolean For the general definition of On, see Table 3.4.

**Legal values:** true, false

**Errors:** configurationerror, typecheck

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

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

**Errors:** rangecheck, typecheck

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

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

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

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

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

A configuration error is generated if setting the Protocol or Interpreter parameter would produce either of the following situations when the Enabled parameter is true:

- **Protocol** with a value of /Raw and **Interpreter** with a value of /PostScript.
- **Protocol** with a value of /Normal and **Interpreter** with a value other than /PostScript.
- **Protocol** with a value of /Normal and **Interpreter** with a value of /AutoSelect.
That is, PostScript language jobs cannot be executed over a channel using the \texttt{/Raw} protocol, and emulators cannot be executed over a channel using the \texttt{/Normal} protocol. Likewise, when doing automatic selection of interpreters and emulators, the \texttt{/Normal} protocol cannot be used.

Either \texttt{Protocol} or \texttt{Interpreter} or both can be set without a password if no other parameters are specified in the execution of \texttt{setdevparams}.

\textbf{Legal values:} /Binary, /Normal, /Raw, /TBCP

\textbf{Errors:} configurationerror, rangecheck, typecheck

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

\textbf{Legal values:} 1, 2

\textbf{Errors:} rangecheck, typecheck

\textbf{Type} name \texttt{(Read-only)} This parameter always has a value of /Communications. For the general definition of \texttt{Type}, see Table 3.3.

\textbf{Legal value:} /Communications

\textbf{Errors:} none

### Parallel Communication Parameters

Table 3.6 lists those parameters typically found in the device parameter sets named \texttt{%Parallel%}, \texttt{%ParallelB%}, \texttt{%ParallelC%}, and so on.

\begin{table}[h]
\centering
\begin{tabular}{lll}
\textbf{Key} & \textbf{Type} & \textbf{Semantics} \\
\hline
DelayedOutputClose & boolean & For the general definition of \texttt{DelayedOutputClose}, see Table 3.4. \\
\textbf{Note} & \texttt{DelayedOutputClose should only be present in this set if bidirectional communication is possible.} & \\
\textbf{Legal values:} & \textit{true, false} & \\
\textbf{Errors:} & none & \\
\end{tabular}
\caption{Parameters present in parallel communication parameter sets}
\end{table}
### Enabled

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
<th>Legal Values</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>For the general definition of Enabled, see Table 3.4.</td>
<td>true, false</td>
<td>configurationerror, typecheck</td>
</tr>
</tbody>
</table>

### Handshake

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
<th>Legal Values</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>This parameter indicates the hardware/software signal interface that is to be used for communications across the parallel (or Centronics) interface. If this key is not present, the default is unidirectional Centronics.</td>
<td>0</td>
<td>configurationerror, rangecheck</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Legal values: 0, 1

### HasNames

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
<th>Legal Values</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>(Read-only) This parameter always has a value of false. For the general definition of HasNames, see Table 3.4.</td>
<td>false</td>
<td>none</td>
</tr>
</tbody>
</table>

### Interpreter

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
<th>Legal Values</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>For the general definition of Interpreter, see Table 3.4.</td>
<td>/PostScript, /AutoSelect, /Diablo630, /EpsonFX850, /HP7475A, /LaserJetIII, /LaserJetLP, /ProprinterXL</td>
<td>configurationerror, rangecheck, typecheck</td>
</tr>
</tbody>
</table>

### On

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
<th>Legal Values</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>For the general definition of On, see Table 3.4.</td>
<td>true, false</td>
<td>configurationerror, typecheck</td>
</tr>
</tbody>
</table>

### OutputDevice

<table>
<thead>
<tr>
<th>Type</th>
<th>Definition</th>
<th>Legal Values</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>This parameter specifies which communications device is to be used for stdout and stderr. If the value of OutputDevice is the empty string, stdout and stderr information is forfeited.</td>
<td>%Serial%, %SerialB%, %SerialC%, etc., %Parallel%, %ParallelB%, %ParallelC%, etc., or the empty string</td>
<td>rangecheck</td>
</tr>
</tbody>
</table>

Errors: rangecheck
Protocol name

For the general definition of Protocol, see Table 3.5.

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

**Errors:** configurationerror, rangecheck, typecheck

**Type**

*(Read-only)* This parameter always has a value of /Communications. For the general definition of Type, see Table 3.3 on page 51.

**Legal value:** /Communications

**Errors:** none

---

**Scsi Communication Parameters**

Table 3.7 lists those parameters typically found in the device parameter sets named %ScsiComm%, %ScsiCommB%, %ScsiCommC%, and so on.

**Table 3.7 Parameters present in SCSI communication parameter sets**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus</td>
<td>string</td>
<td><em>(Read-only)</em> This parameter designates which Scsi bus device parameter set is associated with this %ScsiComm% channel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> %Scsi%, %ScsiB%, %ScsiC%, etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
<tr>
<td>DelayedOutputClose</td>
<td>boolean</td>
<td>For the general definition of DelayedOutputClose, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
<tr>
<td>Enabled</td>
<td>boolean</td>
<td>For the general definition of Enabled, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> configurationerror, typecheck</td>
</tr>
<tr>
<td>HasNames</td>
<td>boolean</td>
<td><em>(Read-only)</em> This parameter always has a value of false. For the general definition of HasNames, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal value:</strong> false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
</tbody>
</table>
Interpreter name For the general definition of Interpreter, see Table 3.4.

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

Errors: configurationerror, rangecheck, typecheck

On boolean For the general definition of On, see Table 3.4.

Legal values: true, false

Errors: configurationerror, typecheck

Type (Read-only) This parameter always has a value of /Communications. For the general definition of Type, see Table 3.3 on page 51.

Legal value: /Communications

Errors: none

LocalTalk Communication Parameters

Table 3.8 lists those parameters typically found in the device parameter sets named %LocalTalk%, %LocalTalkB%, %LocalTalkC%, and so on.

Table 3.8 Parameters present in LocalTalk communication parameter sets

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>DelayedOutputClose</td>
<td>boolean</td>
<td>For the general definition of DelayedOutputClose, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>Enabled</td>
<td>boolean</td>
<td>For the general definition of Enabled, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: configurationerror, typecheck</td>
</tr>
<tr>
<td>Filtering</td>
<td>name</td>
<td>This parameter indicates whether the input stream needs further filtering</td>
</tr>
<tr>
<td></td>
<td></td>
<td>before the data can be correctly interpreted as a page description language.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/InterpreterBased: In this mode, the input stream is filtered as necessary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to conform to the language. For example, the data stream may have been sent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to the printer encoded as a TBCP PostScript language job and must be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>decoded to a normal PostScript language job before it is passed to the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>interpreter (see Protocol in Table 3.5 for a description of TBCP).</td>
</tr>
</tbody>
</table>
/None: Pass the data unchanged to the interpreter.

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

Legal values:  /InterpreterBased, /None

Errors:  configurationerror, rangecheck, typecheck

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

Legal value:  false

Errors:  none

Interpreter  name  For the general definition of Interpreter, see Table 3.4.

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

Errors:  configurationerror, rangecheck, typecheck

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

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

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

Errors:  limitcheck, typecheck

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

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

**Errors:** none

**On** boolean For the general definition of On, see Table 3.4.

**Legal values:** true, false

**Errors:** configurationerror, typecheck

**Type** name *(Read-only)* This parameter always has a value of /Communications. For the general definition of Type, see Table 3.3 on page 51.

**Legal value:** /Communications

**Errors:** none

---

**EtherTalk Communication Parameters**

Table 3.9 lists those parameters typically found in the device parameter sets named %EtherTalk%, %EtherTalkB%, %EtherTalkC%, and so on.

**Table 3.9 Parameters present in EtherTalk communication parameter sets**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>DelayedOutputClose</td>
<td>boolean</td>
<td>For the general definition of DelayedOutputClose, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
<tr>
<td>Enabled</td>
<td>boolean</td>
<td>For the general definition of Enabled, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> configurationerror, typecheck</td>
</tr>
</tbody>
</table>
**EthernetAddress**

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

**Legal values:** A string of 17 characters representing a legal Ethernet address.

**Errors:** none

**EtherTalkType**

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

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

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

**Errors:** typecheck

**EtherTalkZone**

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

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

**Errors:** typecheck

**Filtering**

*name* This parameter indicates whether the input stream needs further filtering before the data can be correctly interpreted as a page description language. For the general definition of *Filtering*, see Table 3.8.

**Warning** In a normal network environment, *Filtering* should be set to `/None` or you will encounter communication problems.

**Legal values:** `/InterpreterBased`, `/None`

**Errors:** configurationerror, rangecheck, typecheck

**HasNames**

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

**Legal value:** `false`

**Errors:** none
### Interpreter

**name**

For the general definition of **Interpreter**, see Table 3.4.

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

**Errors:** configurationerror, rangecheck, typecheck

### On

**boolean**

For the general definition of **On**, see Table 3.4.

**Legal values:** true, false

**Errors:** configurationerror, typecheck

### Type

**name** *(Read-only)*

This parameter always has a value of /Communications. For the general definition of **Type**, see Table 3.3 on page 51.

**Legal value:** /Communications

**Errors:** none

---

### Node Address

Before listing the various network communication parameter sets, we must define the term **node address**. A node address is a unique address for a node on some network. The node address is in the form appropriate for the protocol being used to communicate with the node. The following table lists the various forms that a node address can take.

**Table 3.10 Node Address Forms**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Node Address Form</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/IP</td>
<td>N.N.N.N</td>
<td>where each N is a decimal number in the range 0 to 255</td>
</tr>
<tr>
<td>Novell SPX/IPX</td>
<td>XXXXXXXX:xxxxxxxxxxxxxxx</td>
<td>where each X and each x is a hexadecimal digit in the range 0 to F (upper or lower case) XXXXXXXX is the network part of the address and xxxxxxxxxxx is the Media Access Control part known as the Novell Node Number</td>
</tr>
<tr>
<td>AppleTalk DDP</td>
<td>N.N.n</td>
<td>where each N and each n are decimal numbers in the range of 0 to 255 N.N represent the network part of the address and n represents the node ID</td>
</tr>
</tbody>
</table>

---

### Application Layer Communications Parameters

This section describes those device parameter sets which correspond to the application protocol layer. This is the layer to which the PostScript interpreter (or language emulator) attaches for the purpose of receiving jobs.
and sending data back to the host. Certain parameter sets have been defined by Adobe Systems, Inc. which allow for the use of the TCP/IP protocol over Ethernet. These are LPR, AppSocket, and Telnet. There are also parameters sets associated with the Novell Netware® application layer. They are RemotePrinter and PrintServer.

**LPR**

The UNIX command lpr has the effect of sending a printer job to a printer. On the printer side, the LPR device name is used as the job source for incoming lpr jobs. There is an LPR device parameter set and it is described in Table 3.9. Port 515 is used for LPR. Because LPR (or lpr daemon) is by definition unidirectional, any stdout or stderr information is transmitted by means of the Syslog facility described below. The LPR service depends upon the TCP/IP protocol.

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enabled</strong></td>
<td>boolean</td>
<td>For the general definition of Enabled, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> configurationerror, typecheck</td>
</tr>
<tr>
<td><strong>Filtering</strong></td>
<td>name</td>
<td>This parameter indicates whether the input stream needs further filtering before the data can be correctly interpreted as a page description language. For the general definition of Filtering, see Table 3.8.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Warning</strong> In a normal network environment, Filtering should be set to /None or you will encounter communication problems.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> /InterpreterBased, /None</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> configurationerror, rangecheck, typecheck</td>
</tr>
<tr>
<td><strong>HasNames</strong></td>
<td>boolean</td>
<td>(Read-only) This parameter always has a value of false. For the general definition of HasNames, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal value:</strong> false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
<tr>
<td><strong>Interpreter</strong></td>
<td>name</td>
<td>For the general definition of Interpreter, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> /PostScript, /AutoSelect, /Diablo630, /EpsonFX850, /HP7475A, /LaserJetIII, /LaserJetIIP, /Proprietary</td>
</tr>
</tbody>
</table>
On  boolean  For the general definition of On, see Table 3.4.

Legal values:  true, false

Errors:  configurationerror, typecheck

PrintHost  string  This parameter is a list of at most 2 IP addresses for hosts that have access to the unit; an empty string gives unrestricted access.

Legal values:  An empty string or a string (of 31 or fewer non-null characters) which specifies up to 2 IP addresses separated by the ASCII blank character. An IP address can be of the form N.N.N.N where each N is a decimal number in the range 0 to 255. IP addresses cannot be set to illegal values (e.g., trying to use an IP address equal to 0.0.0.0, 127.0.0.0, 255.255.255.255, N.N.255 or other illegal values will result in a rangecheck error).

Errors:  typecheck, limitcheck, rangecheck

ReceiveWindowSize  integer  Specifying the receive window size is a means of tuning the code for optimal throughput. This setting is enacted at boot time, when memory is allocated for use by the network communications software. The actual window size is established when the connection is opened and may be smaller than this parameter states in order to accommodate the host’s expectations. The receive window size specified here overrides any request for this parameter in the associated sets of type /Parameters, for example, %TCP%.

Legal value:  An integer in the range from 1024 to 59392.

Errors:  typecheck, rangecheck

SendWindowSize  integer  Specifying the send window size is a means of tuning the code for optimal throughput. This setting is enacted at boot time, when memory is allocated for use by the network communications software. The actual window size is established when the connection is opened and may be smaller than this parameter states in order to accommodate the host’s expectations. The send window size specified here overrides any request for this parameter in the associated sets of type /Parameters, for example, %TCP%.

Legal value:  An integer in the range from 1024 to 59392.

Errors:  typecheck, rangecheck
**Type**

*(Read-only)* This parameter always has a value of `/Communications`. For the general definition of **Type**, see Table 3.3 on page 51.

**Legal value:** `/Communications`

**Errors:** none

---

**AppSocket**

AppSocket was created to support TranScript software from Adobe Systems, Inc. It provides a more robust interface than LPR because it utilizes bidirectional communications directly. The AppSocket protocol can be used by drivers other than TranScript and for transmitting data other than PostScript language jobs.

**Table 3.12 Parameters present in the `%AppSocket% communication parameter set**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ControlPortNumber</strong></td>
<td>integer</td>
<td>This parameter denotes a port used by the unit for the purpose of hand-shaking between the host and the unit while setting up a session. A session with the printer prevents other hosts from being able to interrupt the printer to run other jobs. Communication is via TCP, not UDP. The suggested default value is 9101. <strong>Legal values:</strong> A positive integer representing a port number not reserved by any of the standard services. <strong>Errors:</strong> typecheck, rangecheck, configurationerror</td>
</tr>
<tr>
<td><strong>DataPortNumber</strong></td>
<td>integer</td>
<td>This parameter denotes a bidirectional port for transmission of printer language jobs. The suggested default value is 9100. Users are free to use another port number to avoid a conflict with another unit on the network already using 9100. <strong>Legal values:</strong> A positive integer representing a port number not reserved by any of the standard services. <strong>Warning</strong> An error is not raised when a port number is specified that has been previously reserved for some other purpose. <strong>Errors:</strong> typecheck, rangecheck, configurationerror</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DelayedOutputClose</td>
<td>boolean</td>
<td>For the general definition of <strong>DelayedOutputClose</strong>, see Table 3.4.</td>
</tr>
<tr>
<td>Enabled</td>
<td>boolean</td>
<td>For the general definition of <strong>Enabled</strong>, see Table 3.4.</td>
</tr>
<tr>
<td>Filtering</td>
<td>name</td>
<td>This parameter indicates whether the input stream needs further filtering before the data can be correctly interpreted as a page description language. For the general definition of <strong>Filtering</strong>, see Table 3.8.</td>
</tr>
<tr>
<td>HasNames</td>
<td>boolean</td>
<td><em>(Read-only)</em> This parameter always has a value of false. For the general definition of <strong>HasNames</strong>, see Table 3.4.</td>
</tr>
<tr>
<td>Interpreter</td>
<td>name</td>
<td>For the general definition of <strong>Interpreter</strong>, see Table 3.4.</td>
</tr>
<tr>
<td>On</td>
<td>boolean</td>
<td>For the general definition of <strong>On</strong>, see Table 3.4.</td>
</tr>
</tbody>
</table>
**PrintHost**  
string  
This parameter is a single node address (refer to Table 3.10) per protocol for a host that has access to the unit; an empty string gives unrestricted access. Trying to set this parameter to an illegal node address results in a **rangecheck** error.

**Legal values:** An empty string or a string (of 49 or fewer non-null characters) which specifies up to one node address per protocol separated by the ASCII blank character.

**Errors:** typecheck, rangecheck, limitcheck

**ReceiveWindowSize**  
integer  
Specifying the receive window size is a means of tuning the code for optimal throughput. This settings is enacted at boot time, when memory is allocated for use by the network communications software. The actual window size is established when the connection is opened and may be smaller than this parameter states in order to accommodate the host’s expectations. The receive window size specified here overrides any request for this parameter in the associated sets of type /Parameters, for example, %TCP%.

**Legal value:** An integer in the range from 1024 to 59392.

**Errors:** typecheck, rangecheck

**SendWindowSize**  
integer  
Specifying the send window size is a means of tuning the code for optimal throughput. This settings is enacted at boot time, when memory is allocated for use by the network communications software. The actual window size is established when the connection is opened and may be smaller than this parameter states in order to accommodate the host’s expectations. The send window size specified here overrides any request for this parameter in the associated sets of type /Parameters, for example, %TCP%.

**Legal value:** An integer in the range from 1024 to 59392.

**Errors:** typecheck, rangecheck

**StatusPortNumber**  
integer  
This parameter denotes a port used by the unit for the purpose of sending status information back to the host. When using TCP/IP, communication is via UDP, not the TCP transport layer. The suggested default value is 9101. Users may use another port number to avoid a conflict with another unit on the network already using 9101.

**Legal value:** A positive integer representing a port number not reserved by any of the standard services.
Warning An error is not raised when a port number is specified that has been previously reserved for some other purpose.

**Errors:** typecheck, rangecheck, configurationerror

**Type** name (Read-only) This parameter always has a value of /Communications. For the general definition of **Type**, see Table 3.3 on page 51.

**Legal value:** /Communications

**Errors:** none

---

**Telnet**

Telnet gives network users interactive access to and exclusive use of the PostScript interpreter (or emulator). Port 23 is used for Telnet. Telnet is a TCP/IP network service.

**Table 3.13 Parameters present in the %Telnet% communication parameter set**

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

---

3.5 Device Parameters 79
On  boolean  For the general definition of On, see Table 3.4.

Legal values: true, false

Errors: configurationerror, typecheck

Type  (Read-only) This parameter always has a value of /Communications. For the general definition of Type, see Table 3.3 on page 51.

Legal value: /Communications

Errors: none

Novell’s Remote Printer

This is an application that is managed by a Novell Print Server and takes print jobs downloaded from a Print Server. A Remote Printer may be shared by many Print Servers.

Table 3.14 Parameters present in the %RemotePrinter% communication parameter set

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>DelayedOutputClose</td>
<td>boolean</td>
<td>For the general definition of DelayedOutputClose, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>Enabled</td>
<td>boolean</td>
<td>For the general definition of Enabled, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: configurationerror, typecheck</td>
</tr>
<tr>
<td>Filtering</td>
<td>name</td>
<td>For the general definition of Filtering, see Table 3.8. Filtering allows for transport of data not initially intended for this network protocol. For example, if the host environment thought it was transmitting data over a parallel interface, such as LPT of a PC, and the printer is connected through a network interface, Filtering would need to be set to /InterpreterBased.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: /InterpreterBased, /None</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: configurationerror, rangecheck, typecheck</td>
</tr>
</tbody>
</table>
HasNames  boolean  (Read-only) This parameter always has a value of false. For the general definition of HasNames, see Table 3.4.

Legal value: false

Errors: none

Interpreter  name  For the general definition of Interpreter, see Table 3.4.

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

Errors: configurationerror, rangecheck, typecheck

On  boolean  For the general definition of On, see Table 3.4.

Legal values: true, false

Errors: configurationerror, typecheck

Type  (Read-only) This parameter always has a value of /Communications. For the general definition of Type, see Table 3.3 on page 51.

Legal value: /Communications

Errors: none

---

**Novell’s Print Server**

This is the application that communicates with Novell File Servers to download print jobs from the print queues. A Print Server may communicate with multiple File Servers and access multiple print queues.

**Table 3.15 Parameters present in the %PrintServer% communication parameter set**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>DelayedOutputClose</td>
<td>boolean</td>
<td>For the general definition of DelayedOutputClose, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>Enabled</td>
<td>boolean</td>
<td>For the general definition of Enabled, see Table 3.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: configurationerror, typecheck</td>
</tr>
</tbody>
</table>
Filtering
name
For the general definition of **Filtering**, see Table 3.8. **Filtering** allows for transport of data not initially intended for this network protocol. For example, if the host environment thought it was transmitting data over a parallel interface, such as LPT of a PC, and the printer is connected through a network interface, **Filtering** would need to be set to /InterpreterBased.

**Legal values:** /InterpreterBased, /None

**Errors:** configurationerror, rangecheck, typecheck

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

**Legal value:** false

**Errors:** none

Interpreter
name
For the general definition of **Interpreter**, see Table 3.4.

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

**Errors:** configurationerror, rangecheck, typecheck

LoginPassword
string This string parameter specifies the password that the PrintServer uses to gain access to the job queue. Setting this parameter to the empty string indicates that no password has been specified. The value of this parameter returned by the **currentdevparams** operator is the string “(INVALID)” regardless of what the password is set to. Attempts to set the **LoginPassword** to the string “(INVALID)” will be ignored.

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

**Errors:** limitcheck, typecheck

On
boolean For the general definition of **On**, see Table 3.4.

**Legal values:** true, false

**Errors:** configurationerror, typecheck

Type
*(Read-only)* This parameter always has a value of /Communications. For the general definition of **Type**, see Table 3.3 on page 51.

**Legal value:** /Communications

**Errors:** none
Communications Parameter Sets Which Correspond to Various Network Services

This section covers the device parameter sets of type /Parameters that control various network services. The services are known as SNMP and Syslog.

SNMP

SNMP, Simple Network Management Protocol, provides a means for the system administrator to query for information about the unit. The information that can be queried is driven by a database called a Management Information Base (MIB). Refer to “A Simple Network Management Protocol” for details about SNMP⁴. It is not a communications port for PostScript language jobs, thus, the parameter set is of type /Parameters. The parameters listed in Table 3.16 are those SNMP parameters which need to be accessible from the PostScript language. These are the only parameters which are changeable from an environment separate from SNMP (the network side). The rules about when changes take effect to each parameter within this parameter set are described in Table 3.16.

Table 3.16 Parameters present in the %SNMP% parameter set

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrivateHost</td>
<td>string</td>
<td>This parameter is a single node address (refer to Table 3.10) per protocol of a host that is able to set those SNMP variables that can be written; an empty string indicates that no host has access. The empty string is the usual default value so that the unit will need to have this parameter explicitly set via the PostScript operator setdevparams prior to using SNMP. Changes to this parameter do not take effect until the unit is re-initialized.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: An empty string or a string (of 49 or fewer non-null characters) which specifies up to one node address per protocol separated by the ASCII blank character.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: typecheck, rangecheck, limitcheck</td>
</tr>
<tr>
<td>SysContact</td>
<td>string</td>
<td>The convention is to use the name and phone number or address of the person responsible for the unit. Changes to this parameter take effect immediately.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal value: A string of 32 or fewer non-null characters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: typecheck, limitcheck</td>
</tr>
</tbody>
</table>

**SysLocation**  
string  
Location of the raster output device unit. Changes to this parameter take effect immediately.

**Legal value:** A string of 32 or fewer non-null characters.

**Errors:** typecheck, limitcheck

**SysName**  
string  
Name of the raster output device unit (expected by SNMP). Changes to this parameter take effect immediately.

**Legal value:** A string of 32 or fewer non-null characters.

**Errors:** typecheck, limitcheck

**TrapHost**  
string  
This parameter is a list of one or more `<node address/community>` pairs for each protocol which has a host that is able to receive traps. Refer to Table 3.10 for the syntax of a node address. A slash is used as a delimiter between the `node address` and the `community` string. The ASCII blank is used to separate each pair in the list. The `community` string portion is case insensitive. An empty string indicates that no host is sent traps. Here are some example `community` strings:

- public
- proxy
- private
- regional
- core

For example, the value (130.248.224.46/public) is an IP address for a trap host node in a *public* community.

The empty string is the usual default value so that the unit will need to have this parameter explicitly set via the PostScript operator `setdevparams` prior to using the trap host facility. Changes to this parameter do not take effect until the unit is re-initialized.

**Legal values:** An empty string or a string (of 65K or fewer non-null characters) which specifies one or more `<node address/community>` pairs separated by the ASCII blank character.

**Errors:** typecheck, rangecheck, limitcheck
Type  (Read-only) This parameter always has a value of /Parameters. For the general definition of Type, see Table 3.3 on page 51.

Legal value: /Parameters

Errors: none

Syslog

Syslog is a logging facility back to a UNIX host. The LogPriority value indicates which log messages will be seen by the host. Much of the logging deals with network-specific information, but may include any other pertinent information the unit wishes to convey. Communication for %Syslog% is via the UDP (User Datagram Protocol) transport layer. Changes to this parameter set do not take effect until the unit is re-initialized.

Table 3.17 Parameters present in the %Syslog% parameter set

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogHost</td>
<td>string</td>
<td>This string contains an IP address for a host that receives Syslog messages from the unit; an empty string indicates that no Syslog messages are to be sent by the unit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> An empty string or a string (of 15 or fewer non-null characters) which specifies a legal IP address. An IP address is of the form N.N.N.N where each N is a decimal number in the range 0 to 255. Trying to use an IP address equal to 0.0.0.0, 255.255.255.255 or N.N.N.255 will result in a rangecheck error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> typecheck, limitcheck</td>
</tr>
<tr>
<td>LogPriority</td>
<td>integer</td>
<td>This parameter designates which logging messages are to be sent on to the Syslog host. All logging messages associated with the specified LogPriority and those of higher priority (smaller numbers are higher priority) are sent. The following is a list, from the BSD and SunOS convention, of priorities and their corresponding meaning.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0 – unit is no longer usable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1 – messages indicating immediate action is needed on the part of a system administrator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2 – critical error messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3 – error messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4 – warning messages</td>
</tr>
</tbody>
</table>
• 5 – normal but significant conditions
• 6 – informational messages
• 7 – debugging messages

Legal values: An integer in the range from 0 to 7.

Errors: typecheck, rangecheck

Type

(Read-only) This parameter always has a value of /Parameters. For the general definition of Type, see Table 3.3 on page 51.

Legal value: /Parameters

Errors: none

Communications Parameter Sets Which Correspond to Lower Protocol Layers

This section describes the device parameter sets of type /Parameters that control the transport, network, data link and physical layers of the TCP/IP protocol services.

TCP

TCP stands for Transmission Control Protocol and is the transport layer which is responsible for reliable data transfer by guaranteeing message delivery and reception. It is a connection-oriented protocol. If a packet is lost, it will be retransmitted. Changes to parameters in the TCP set do not take effect until the unit is re-initialized.

Table 3.18 Parameters present in the %TCP% parameter set

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>boolean</td>
<td>A value of true means that the TCP protocol is activated at boot time. Otherwise it is off.</td>
</tr>
</tbody>
</table>

Legal values: true, false

Errors: configurationerror, typecheck
ReceiveWindowSize

integer

Specifying the receive window size is a means of tuning the code for optimal throughput. This setting is enacted at boot time, when memory is allocated for use by the network communications software. The actual window size is established when the connection is opened and may be smaller than this parameter states in order to accommodate the host’s expectations.

**Legal values:** An integer in the range from 1024 to 59392.

**Errors:** typecheck, rangecheck

SendWindowSize

integer

Specifying the send window size is a means of tuning the code for optimal throughput. This setting is enacted at boot time, when memory is allocated for use by the network communications software. The actual window size is established when the connection is opened and may be smaller than this parameter states in order to accommodate the host’s expectations.

**Legal values:** An integer in the range from 1024 to 59392.

**Errors:** typecheck, rangecheck

Type

*(Read-only)* This parameter always has a value of `/Parameters`. For the general definition of Type, see Table 3.3 on page 51.

**Legal value:** `/Parameters`

**Errors:** none

User Datagram Protocol (UDP)

UDP is a connectionless (or datagram) protocol used in the TCP/IP networking suite. When using UDP with a peer host, there is no need for “handshaking” prior to communication. UDP packets are sent without any guarantee of delivery and may arrive at the destination in any order.

Table 3.19  *Parameters present in the %UDP% parameter set*

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checksum</td>
<td>boolean</td>
<td>This boolean specifies whether checksum values will be inserted in outgoing packets formed by the software. The default should be <em>true</em>.</td>
</tr>
</tbody>
</table>

**Legal values:** true, false

**Errors:** configurationerror, typecheck
On  boolean  A value of *true* means that the UDP protocol is activated at boot time. Otherwise it is off.

**Legal values:**  true, false

**Errors:**  configurationerror, typecheck

**Type**  (Read-only) This parameter always has a value of /Parameters. For the general definition of **Type**, see Table 3.3 on page 51.

**Legal value:**  /Parameters

**Errors:**  none

**IP**

IP stands for *Internet Protocol* and is the network layer which is responsible for routing messages to their destinations. This layer decides which physical interface is to send outgoing messages and which transport layer is to receive incoming message. Changes to parameters in the IP set do not take effect until the unit is re-initialized.

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
</table>
| BroadcastAddress| string   | This parameter is the broadcast address mask used when broadcasting messages to the local network. **BroadcastAddress** shall reflect the current broadcast address mask in use by the unit. In order to “set” the **BroadcastAddress** explicitly and have it take effect the next time the unit is initialized, you must have **IPAddressDynamic** set to *false* when issuing setdevparams.  

**Note**  If the **BroadcastAddress** is not legal with respect to the **IPAddress** and **NetworkMask** given, it shall be changed to a value that is legal with no warning to the user. For example, suppose the **IPAddress** is 134.14.15.16 and the **BroadcastAddress** is 134.14.255.255. If the user changes **IPAddress** to 134.15.15.16 without explicitly changing the **BroadcastAddress**, the **BroadcastAddress** shall automatically be changed to 134.15.255.255.  

**Legal values:**  A string of 15 or fewer non-null characters which specifies a legal **BroadcastAddress**. A **BroadcastAddress** is of the form *N.N.N.N* where each *N* is a decimal number in the range 0 to 255.

**Errors:**  limitcheck, typecheck, rangecheck
**GatewayAddress**

This parameter contains the destination address/gateway address pairs to other networks. The empty string specifies that dynamic routing, if available in the product, be enabled. In accordance with the route command, a GatewayAddress is fully defined here as a destination address and a gateway address. The syntax is <destination address/gateway address>, using the slash as delimiter between the two subfields. Multiple address pairs can be specified and are separated by a space delimiter. In accordance with the route command, a destination address is defined as specifying the network address. A valid network address varies according to the Class. A Class A network address requires the first field be non-zero. The others may be zero (for a net with no subnets), or contain subnet address information. A Class B network address requires the first two fields to be non-zero. A Class C network address requires the first three fields to be non-zero. A network address of 0.0.0.0 is a special case used by default if no previous network address matches the desired target IP address. If multiple entries have the same address, then the earlier entry will be ignored. GatewayAddress shall reflect the current <destination address/gateway address> pairs to other networks in use by the unit. The GatewayAddress parameter may be “set” explicitly at any time. It will only take effect upon unit initialization, and then only if the IPAddressDynamic parameter is set to false via setdevparams. If IPAddressDynamic is set to true at unit initialization time, the GatewayAddress parameter will not take effect. The routing information will be gathered via dynamic routing using a RIP request to the network. The default should be the empty string, implying dynamic routing.

**Legal values:** An empty string or a string (of 65K or fewer non-null characters) which specifies one or more legal <destination address/gateway address> pairs. These addresses are Internet Protocol addresses of the form N.N.N.N where each N is a decimal number in the range 0 to 255. Loopback addresses (127.N.N.N) and broadcast addresses (N.N.255) are illegal for either the destination or gateway part of the pair.

**Errors:** limitcheck, typecheck, rangecheck

**IPAddress**

This is a unique string which represents the Internet Protocol address of the unit. The Internet Protocol address is mapped directly to the lowest physical address by which the unit is known (e.g., EthernetAddress if Physical is %EthernetPhysical%). IPAddress shall reflect the current IP address in use by the unit. In order to “set” the IPAddress explicitly and have it take effect the next time the unit is initialized, you must have IPAddressDynamic set to false when issuing setdevparams. The default should be an empty string which implies that the IP protocol layer is not active.
Note Whenever IPAddressDynamic is true, currentdevparams will return a value for the parameter IPAddress that has been determined by a BOOTP or RARP sequence during boot up. Changing the IPAddress parameter to some other value via setdevparams has the effect of changing the user explicit value which is only used if IPAddressDynamic is false. currentdevparams will return the user explicit value of IPAddress only when IPAddressDynamic is false.

Legal values: An empty string or a string (of 15 or fewer non-null characters) which specifies a legal IP address. An IP address is of the form N.N.N.N where each N is a decimal number in the range 0 to 255 (trying to set an IP address equal to 0.0.0.0, 255.255.255.255, 127.N.N.N, N.N.0, N.N.255 or any address whose first field is in the range 224 to 255 will result in a rangecheck error).

Errors: limitcheck, typecheck, rangecheck

IPAddressDynamic boolean A value of true indicates that the IPAddress is obtained by a BOOTP or RARP sequence during boot up. The value of false means that the IPAddress must be explicitly set by a PostScript language job via setdevparams in order for connections to be made on the local network. The default value is usually false.

Legal values: true, false

Errors: typecheck

NetworkMask string This parameter indicates which fields of the IPAddress designate the network portion of the IP address and which designate the node portion. For example, the value 255.255.255.0 is a NetworkMask for a Class B network with subnets. The NetworkMask is used to determine if a certain IP address is on the same network as the unit. NetworkMask shall reflect the current network mask in use by the unit. In order to “set” the NetworkMask explicitly and have it take effect the next time the unit is initialized, you must have IPAddressDynamic set to false when issuing setdevparams.

Note If NetworkMask is set to a value that is not legal when compared to the IPAddress, the NetworkMask shall be changed to a value that is legal with no warning to the user. For example, if a Class B IPAddress is given with a Class A network mask, the NetworkMask shall be changed to the default Class B network mask. The default Class A network mask is 255.0.0.0. The
The default Class B network mask is 255.255.0.0. The default Class C network mask is 255.255.255.0. No subnets are accounted for in these default network masks.

**Legal values:** A string of 15 or fewer non-null characters which specifies a legal IP mask. IP masks are of the form $N.N.N.N$ where each $N$ is a decimal number in the range 0 to 255.

**Errors:** limitcheck, typecheck

<table>
<thead>
<tr>
<th><strong>On</strong></th>
<th>boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>A value of <em>true</em> means that the IP protocol layer is activated at boot time. Otherwise it remains off.</td>
<td></td>
</tr>
</tbody>
</table>

**Legal values:** true, false

**Errors:** configurationerror, typecheck

**Physical** string *(Read-only)* This parameter specifies the physical layer over which IP is accessed. The string is set to a device parameter set corresponding to a physical communications medium, such as the string (%EthernetPhysical%). A network layer parameter set can be associated with one and only one physical layer parameter set by the **Physical** parameter.

**Legal values:** A string of 32 or fewer non-null characters which specifies a physical layer.

**Errors:** none

**TransmitEncapsulation name** This parameter specifies the transmit encapsulation type. The value /802.3-2-SNAP indicates IEEE 802.3 MAC header followed by a 802.2 LLC and SNAP header. The value /DIX indicates Ethernet Version II. The default should be /DIX for Ethernet.

**Legal values:** /802.3-2-SNAP, /DIX

**Errors:** typecheck

**Type** *(Read-only)* This parameter always has a value of /Parameters. For the general definition of **Type**, see Table 3.3 on page 51.

**Legal value:** /Parameters

**Errors:** none
**Sequenced Packet Exchange (SPX) Protocol**

SPX is the Novell Netware® connection oriented protocol. When using SPX to communicate with a peer host, there is “handshaking” before the connection is ready. The delivery of SPX packets are expected to be acknowledged to guarantee delivery and packets arrive in sequence at the destination. Unlike TCP, it does not provide a sliding window functionality for flow control.

**Table 3.21 Parameters present in the %SPX% parameter set**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>On</strong></td>
<td>boolean</td>
<td>A value of <em>true</em> means that the SPX protocol is activated at boot time. Otherwise it is off.</td>
</tr>
<tr>
<td><strong>ReceiveWindowSize</strong></td>
<td>integer</td>
<td>Specifying the receive window size is a means of tuning the code for optimal throughput. This settings is enacted at boot time, when memory is allocated for use by the network communications software. The actual window size is established when the connection is opened and may be smaller than this parameter states in order to accommodate the host’s expectations.</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>(Read-only)</td>
<td>This parameter always has a value of /Parameters. For the general definition of <strong>Type</strong>, see Table 3.3 on page 51.</td>
</tr>
</tbody>
</table>

Legal values:  
* true, false

Errors:  
* configurationerror, typecheck

Errors:  
* typecheck, rangecheck

Legal value:  
/Parameters

Errors:  
* none
**Internetwork Packet Exchange (IPX) Protocol**

IPX is the Novell Netware® connectionless (or datagram) protocol. When using IPX with a peer host, there is no need for “handshaking” prior to communication. IPX packets are sent without any guarantee of delivery and may arrive at the destination in any order. Netware broadcasting is done using IPX.

| Table 3.22 Parameters present in the %IPX% parameter set |
|---|---|---|
| **Key** | **Type** | **Semantics** |
| **Checksum** | boolean | This boolean specifies whether checksum values will be inserted in outgoing packets formed by the software. The default should be true. |
| **Legal values:** | true, false |
| **Errors:** | none |
| **NetworkAddress** | string | (Read-only) This parameter identifies the network in which the unit is located. The concatenation of the NetworkAddress and the Novell Node Number will uniquely identify the unit on the network. The Novell Node Number is derived from the Media Access Control (MAC) address of the networking media. For Ethernet, the Novell Node Number is the EthernetAddress parameter of the %EthernetPhysical% set. The NetworkAddress is obtained from the Novell file server on the local net upon booting the printer. |
| **Legal values:** | An empty string or a string (of 8 or fewer non-null characters) which specifies a legal Novell network address. A Novell network address is of the form XXXXXXXX where each X represents a digit in hexadecimal in the range 0 to F. |
| **Errors:** | none |
| **On** | boolean | A value of true means that the IPX protocol layer is activated at boot time. Otherwise it remains off. |
| **Legal values:** | true, false |
| **Errors:** | configurationerror, typecheck |
Physical string  *(Read-only)* This parameter specifies the physical layer over which IPX is accessed. The string is set to a device parameter set corresponding to a physical communications medium, such as the string (%EthernetPhysical%). A network layer parameter set can be associated with one and only one physical layer parameter set by the Physical parameter.

**Legal values:** A string of 32 or fewer non-null characters which specifies a physical layer.

**Errors:** none

TransmitEncapsulation name  This parameter specifies the transmit encapsulation type. The value /802.3-2 is for specifying an 802.3 header followed by 802.2 header for IEEE-standard-protocols. The value /802.3-X is for specifying an 802.3 header followed by an XNS header (used by IPX only). The value /802.3-2-SNAP indicates IEEE 802.3 MAC header followed by a 802.2 LLC and SNAP header for non-IEEE-standard protocols. The value /DIX indicates Ethernet Version II. The value of /Adaptive indicates that by the nature of the interaction between host and printer, an encapsulation format to use in responses to the host can be derived at boot time. The default should be /802.3-X.

**Legal values:** /802.3-2, /802.3-X, /802.3-2-SNAP, /DIX, /Adaptive

**Errors:** typecheck

Type *(Read-only)* This parameter always has a value of /Parameters. For the general definition of Type, see Table 3.3 on page 51.

**Legal value:** /Parameters

**Errors:** none
**EthernetPhysical**

The `%EthernetPhysical%` device parameter set corresponds to a physical Ethernet connector and its associated hardware and the data link layer software which handles events from this device. Changes to parameters in this set do not take effect until the unit is re-initialized.

**Table 3.23 Parameters present in the `%EthernetPhysical%` parameter set**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectorType</td>
<td>name</td>
<td><em>(Read-only)</em> This parameter indicates which Ethernet connector type is being used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> /RJ45, /BNC, /AUI, /AAUI</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
<tr>
<td>EthernetAddress</td>
<td>string</td>
<td><em>(Read-only)</em> This parameter returns a unique string that represents the Ethernet address of the unit. The string is of the form XX:XX:XX:XX:XX where each X represents a digit in hexadecimal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> When using Novell, the Ethernet address is also known as the “Novell Node Number”.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> A string of 17 characters representing a legal Ethernet address.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
<tr>
<td>Name</td>
<td>string</td>
<td><em>(Read-only)</em> This parameter specifies the mnemonic name, such as (le0) for “Lance chip interface unit 0” or (so0) for “Sonic chip interface unit 0”, for the Ethernet interface used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> Any string of 16 or fewer non-null characters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
<tr>
<td>On</td>
<td>boolean</td>
<td>A value of <em>true</em> means that the Ethernet channel is enabled at boot time. Otherwise it remains off.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> <em>true, false</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> configurationerror, typecheck</td>
</tr>
</tbody>
</table>
**Type**

*(Read-only)* This parameter always has a value of /Parameters. For the general definition of **Type**, see Table 3.3 on page 51.

**Legal value:** /Parameters

**Errors:** none
3.5.3 File System Parameters

Parameters Present in Parameter Sets of Type /FileSystem

Table 3.24 Parameters common to device sets of type /FileSystem

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

**Legal values:** true, false

**Errors:** none

Disk, Cartridge and Rom Parameter Tables

Table 3.25 contains a list of the current disk parameters. Table 3.26 contains a list of the current cartridge or rom parameters. The name “%rom%” is used instead of “%cartridge%” for a cartridge that is non-removable and non-writeable.

*Read-only* refers to their access by language operators (for example, setdevparams, currentdevparams). A read-only parameter can change value but not as the result of invoking setdevparams. Changes to parameters of type /FileSystem take place immediately.

In Table 3.25 and Table 3.26, it should be understood that a page is a unit of storage whose size is file-system dependent.

Table 3.25 Parameters present in disk (/FileSystem) devices

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockSize</td>
<td>integer</td>
<td><em>(Read-only)</em> This parameter indicates the disk/cartridge formatting size of a page (for the logical and physical size of the media). The formatting size of a page for a cartridge is 1 byte per block. The formatting size of a page for a disk using the Adobe File System is 1024 bytes per block.</td>
</tr>
</tbody>
</table>

**Legal values:** Any non-zero positive integer (typically 1024).

**Errors:** none
**Bus**  
String  
*(Read-only)* With Adobe’s storage device implementation, **Bus** will indicate the name of the SCSI bus on which this disk resides. This parameter is in the form of a string that can be used as input to `setdevparams` or `currentdevparams` to get bus parameters.

**Legal values:**  
%Scsi%, %ScsiB%, %ScsiC%, etc.

**Errors:**  
none

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

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

**Errors:**  
none

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

**Legal values:**  
true, false

**Errors:**  
none

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

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

1  
indicates that the current file system (if any) is to be deleted and a new one of size **LogicalSize** created (the media is assumed to have been formatted already). The device must first be mounted; otherwise an *ioerror* will result. For more information, see **LogicalSize**.

2  
reformats the entire media before creating a new file system of size **LogicalSize**. The **Interleave** parameter also plays a role in how the media is to be formatted. See **Interleave** below for details.

3 or greater has the same effect as the value 2 and also carries out product dependent actions, which typically consist of reformattting the disk and running integrity tests before creating the file system. Some devices can have additional parameters that serve as arguments to **InitializeAction**.
Legal values: Any non-negative integer.

Errors: none

Interleave integer The purpose of Interleave is to arrange logically contiguous sectors on the disk in a way that is most efficient for the system using that disk. This parameter is used only when the media is being formatted (refer to InitializeAction, described above).

For example, assume there are 16 sectors going around a single track on a disk. If the first sector has a logical number of 1, the second 2, the third 3, and so on, it is referred to as “1 to 1 interleave” and the value of /Interleave is 1. In this case the system must be very fast in order to be able to take data from the disk, one sector immediately after another. If the system fails to consume the first sector in time for the second sector, the system has to wait an entire revolution of the disk to get the next sector. This can give very poor performance.

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

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

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

Errors: ioerror

LogicalSize integer When set, this parameter specifies the size of the file system to be created and is used as an argument to the action carried out by InitializeAction. If LogicalSize is 0, InitializeAction uses a default size that is normally the size of the entire media within the device. For more information, see InitializeAction.
When queried, this parameter indicates the current size of the file system on the device (in pages were the page size is indicated by the parameter \texttt{BlockSize}). A value of 0 indicates that the device is not mounted.

If \texttt{LogicalSize} is set with a certain value and then the device is reformatted, a query of \texttt{LogicalSize} should return the value that was set. However, if the parameter is queried at any time before the media within the device is reformatted, it may return a different value from what was set because it may return the current size.

\textbf{Legal values:} Any non-negative integer or 0. The value of \texttt{LogicalSize} must be less than or equal to the value of \texttt{PhysicalSize}.

\textbf{Errors:} rangecheck, typecheck

\textbf{Mounted} boolean If this parameter is set to \texttt{true}, the system attempts to mount the device. If set to \texttt{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 \texttt{Mounted} immediately after setting it.

\texttt{Mounted} raises a \texttt{configurationerror} if it is set to \texttt{true} and mounting fails, or if it is set to \texttt{false} and dismounting fails.

\textbf{Legal values:} true, false

\textbf{Errors:} configurationerror, typecheck

\textbf{PhysicalSize} integer (Read-only) This parameter indicates the size of the media (in pages where the page size is indicated by the parameter \texttt{BlockSize}). This parameter is only valid when the device is mounted (that is, \texttt{Mounted} is set to \texttt{true}). A value of 0 indicates that the device is not mounted.

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

\textbf{Errors:} none

\textbf{Removable} boolean (Read-only) This parameter indicates whether the device supports removable media. Depending on how the removable media device operates, setting \texttt{Mounted} to \texttt{false} will either eject the media or allow its removal. When the media has been removed, it cannot be mounted again until it is re-inserted.

\textbf{Legal values:} true, false

\textbf{Errors:} none
Searchable  boolean  This parameter indicates whether the device participates in searches in file system operations that have specified a file name without specifying a device. See section 3.8.2, “Named Files” of the PostScript Language Reference Manual, Second Edition for more information.

Note  Devices that support removable media (on some products) will initially have Searchable be false. Searchable must be explicitly set to true when it is desired to have the media be searched.

Legal values:  true, false

Errors:  none

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

Legal values:  Any non-negative integer.

Errors:  none

Type  name  (Read-only) This parameter always has a value of FileSystem. For the general definition of Type, see Table 3.3 on page 51.

Legal value:  /FileSystem

Errors:  none

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

Legal values:  true, false

Errors:  none

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockSize</td>
<td>integer</td>
<td>(Read-only) For the general definition of BlockSize, see Table 3.25.</td>
</tr>
</tbody>
</table>

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

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

**Legal values:** Any integer.

**Errors:** none

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

**Legal values:** Any integer.

**Errors:** none

**Free** integer *(Read-only)* For the general definition of **Free**, see Table 3.25.

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

**Errors:** none

**HasNames** boolean *(Read-only)* For the general definition of **HasNames**, see Table 3.24.

**Legal values:** true, false

**Errors:** none

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

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

**Legal values:** 0 or 1.

**Errors:** none

**LogicalSize** integer For the general definition of **LogicalSize**, see Table 3.25.

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

**Errors:** rangecheck, typecheck
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mounted</td>
<td>boolean</td>
<td>For the general definition of <strong>Mounted</strong>, see Table 3.25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong></td>
</tr>
<tr>
<td>PhysicalSize</td>
<td>integer</td>
<td>(Read-only) For the general definition of <strong>PhysicalSize</strong>, see Table 3.25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong></td>
</tr>
<tr>
<td>Removable</td>
<td>boolean</td>
<td>(Read-only) For the general definition of <strong>Removable</strong>, see Table 3.25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong></td>
</tr>
<tr>
<td>Searchable</td>
<td>boolean</td>
<td>For the general definition of <strong>Searchable</strong>, see Table 3.25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong></td>
</tr>
<tr>
<td>SearchOrder</td>
<td>integer</td>
<td>For the general definition of <strong>SearchOrder</strong>, see Table 3.25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong></td>
</tr>
<tr>
<td>Type</td>
<td>name</td>
<td>(Read-only) This parameter always has a value of FileSystem. For the general definition of <strong>Type</strong>, see Table 3.3 on page 51.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal value:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong></td>
</tr>
<tr>
<td>Writeable</td>
<td>boolean</td>
<td>(Writeable, but only during a mount) For the general definition of <strong>Writeable</strong>, see Table 3.25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong></td>
</tr>
</tbody>
</table>
3.5.4 %os% Device Parameters

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

Table 3.27 Parameters present in the %os% parameter sets

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasNames</td>
<td>boolean</td>
<td>(Read-only) This parameter always returns a value of true. For the general definition of HasNames, see Table 3.24.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal value: true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>InitializeAction</td>
<td>integer</td>
<td>(Read-only) This parameter always returns a value of 0. For the general definition of InitializeAction, see Table 3.25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal value: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>Mounted</td>
<td>boolean</td>
<td>(Read-only) This parameter always returns a value of true. For the general definition of Mounted, see Table 3.25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal value: true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: typecheck</td>
</tr>
<tr>
<td>Removable</td>
<td>boolean</td>
<td>(Read-only) This parameter always returns a value of false. For the general definition of Removable, see Table 3.25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal value: false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>Searchable</td>
<td>boolean</td>
<td>(Read-only) This parameter always returns a value of true. For the general definition of Searchable, see Table 3.25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>SearchOrder</td>
<td>integer</td>
<td>This parameter initially returns a value of 2. For the general definition of SearchOrder, see Table 3.25.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: Any non-negative integer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>Type</td>
<td>(Read-only) This parameter always returns a value of FileSystem. For the general definition of Type, see Table 3.3 on page 51.</td>
<td>name</td>
</tr>
<tr>
<td>Writeable</td>
<td>(Read-only) This parameter always returns a value of true. For the general definition of Writeable, see Table 3.25.</td>
<td>boolean</td>
</tr>
</tbody>
</table>
3.5.5 Scsi Bus Parameter Set

The parameters in Table 3.28 are used to configure the SCSI bus. The (%Scsi%) parameter set is always present in a printer that has a SCSI bus, even if no devices are present on the SCSI bus. If more than one SCSI bus is present, the first is called %Scsi%, the second %ScsiB%, the third %ScsiC%, and so on. Changes to SCSI parameters do not take effect until the next time the system is initialized.

Table 3.28 Parameters present in the %Scsi% parameter sets

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>BootDelay†</td>
<td>integer</td>
<td>This parameter indicates how long the disk I/O driver should wait (in seconds) during system initialization for the disk to spin up, before determining that a disk is not present or not responding. A value of 0 means that there is no waiting for the disk to spin up. You should set this parameter in accordance with the characteristics of the disk attached to the printer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: Any positive integer or 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>CheckParity</td>
<td>boolean</td>
<td>This parameter indicates if parity on the SCSI bus is to be checked. The default value is usually true.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Warning Setting CheckParity to true on products that do not support parity checking would be unwise. Refer to the individual product addendum for information on whether or not a given product can do this checking.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: none</td>
</tr>
<tr>
<td>InitiatorId†</td>
<td>integer</td>
<td>This parameter is the address on the SCSI bus used by the printer when it serves as initiator. The default value is usually 7.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: A non-negative integer in the range 0 to 7.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors: configurationerror</td>
</tr>
<tr>
<td>Poll†</td>
<td>integer</td>
<td>This parameter is a bit-encoded specification of which addresses on the SCSI bus should be polled by the printer when it looks for disks during system initialization. For example, a 1 in bit 0 means poll for disk0. Any bits in this mask which correspond to addresses that are used as the printer’s InitiatorId or TargetId, as the InitiatorId for other hosts on the bus, or as the TargetId of peripherals belonging to other hosts on the bus should be set to 0 (meaning “do not poll”). If the bit is set to poll the address corresponding to the printer’s InitiatorId or Target Id, a configurationerror is generated. If the bit is set to poll an address that shouldn’t be polled, anomalies may occur.</td>
</tr>
</tbody>
</table>
on the bus. **Poll** is expressed as an integer bit mask in the range 0 to 254 (never 255 since all bits cannot be on — one bit must be reserved for **InitiatorId**). The default value is usually 127 (7F in hexadecimal).

**Legal values:** An integer bit mask in the range 0 to 254.

**Errors:** configurationerror

**TargetId** integer This parameter is the SCSI bus address reserved by the printer for use as the %ScsiComm% communications channel. This address may be the same as the **InitiatorId**.

**Legal values:** A positive integer in the range 0 to 7.

**Errors:** configurationerror

**Type** (Read only) This parameter always has a value of /Parameters. For the general definition of **Type**, see Table 3.3 on page 51.

**Legal value:** /Parameters

**Errors:** none

*= present only when ScsiComm is used on the bus
†= present only when disks are present on the bus
3.5.6 Engine Device Parameters

Table 3.29 lists the parameters associated with the %Engine% device.

Table 3.29  Parameters present in the %Engine% parameter set

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSizeStandard</td>
<td>name</td>
<td>This parameter assists the engine in determining the physical dimensions of the paper when B4 or B5 paper is selected. There are two choices for the value of BSizeStandard:</td>
</tr>
<tr>
<td>ISO: ISO is the abbreviation for the International Standards Organization, which is the international body that defines the “metric” paper sizes (A4, A3, B5, B4, B3, etc.). These are the paper sizes used in Europe and much of the rest of the world. The table below lists the dimensions for the B4 and B5 paper sizes as defined by ISO:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>250 x 353 mm. or 709 x 1001 default units</td>
<td></td>
</tr>
<tr>
<td>B5</td>
<td>176 x 250 mm. or 499 x 709 default units</td>
<td></td>
</tr>
<tr>
<td>JIS: JIS is the abbreviation for the Japanese Institute for Standardization, the national body which specifies standards for use in the country of Japan. Japan also uses the standard “A” paper sizes. However, they use a slightly different definition of the “B” paper sizes. The table below lists the dimensions for the B4 and B5 paper sizes for JIS:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>257 x 364 mm. or 729 x 1032 default units</td>
<td></td>
</tr>
<tr>
<td>B5</td>
<td>182 x 257 mm. or 516 x 729 default units</td>
<td></td>
</tr>
</tbody>
</table>

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

Legal values: ISO, JIS

Errors: rangecheck

Darkness       real  This parameter controls the overall lightness or darkness of the rendered page. This parameter does not affect the frame buffer, nor does it have any computational overhead. Legal values are real numbers from 0.0 through 1.0. A value of 0.0 means minimum darkness, 1.0 means maximum darkness. This option is provided in some products whose marking hardware allows software control of colorant application. The default value is product dependent.

Legal values: Real numbers in the range [0.0 –1.0] inclusive.

Errors: rangecheck
**PageCount** integer  This parameter is a count of all pages fed by the engine. The count includes all of the pages successfully printed as well as the pages that were jammed or spoiled. The value of **PageCount** is determined by querying the engine.

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

**Errors:** typecheck

**TimeToStandby** integer  After the specified number of minutes, the engine will go into a “standby” mode, in which it stops trying to keep itself ready to print a page; that is, it stops keeping its fuser hot. The next time the controller sends a feed or prefeed command, the engine will enter the “warming up” state until it is ready to print. The range of acceptable values for **TimeToStandby** are product-specific. An unallowed value is rounded to the nearest allowed value. Specifying a value of 0 for this parameter has the effect of never letting the printer enter the “standby” mode.

**Legal values:** Product-specific, although typically an integer in the range [0 – 720] inclusive.

**Errors:** rangecheck

**Type** name  *(Read-only)* This parameter always has a value of **Parameters**. For the general definition of **Type**, see Table 3.3 on page 51.

**Legal value:** /Parameters

**Errors:** none
### 3.5.7 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 3.4) 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 for 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:

```
(%Serial%) <</Interpreter /Diablo630 /Protocol /Raw>> setdevparams
```

gives Diablo630-like functionality to input channel Serial on a PostScript printer that has a Diablo630 emulator. This functionality is invoked at the next job boundary.

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

Tables 3.30 through 3.33 describe the parameters for the LaserJetIII emulator, the LaserJetIIP emulator, the color version of the HP7475A plotter emulator, and the Diablo630 emulator.

#### Table 3.30 Parameters present for the LaserJetIII emulator

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copies</td>
<td>integer</td>
<td>This parameter specifies the default number of copies of a document to be printed.</td>
</tr>
<tr>
<td>Duplex</td>
<td>integer</td>
<td>This parameter sets the initial state of duplexing within a PCL job for printers which are capable of duplex operation. Language commands within the print stream can override the setting of this parameter. Acceptable values for <code>Duplex</code> are listed below.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>simplex</td>
</tr>
<tr>
<td>1</td>
<td>long-edge binding duplex</td>
</tr>
<tr>
<td>2</td>
<td>short -edge binding duplex</td>
</tr>
</tbody>
</table>
The default value for **Duplex** is 0, which means that duplexing is not performed.

**FontFixed** boolean If true, a fixed pitch font (e.g., *Courier*) is requested. If it is false, a proportional spaced font is requested.

**FontHeight** integer This is the height of the font, applicable to scalable proportional fonts. This value is a point-size quantity, multiplied by 100 to avoid floating-point representation. A font that is 8.5 points in height would have the value “850” in this parameter. Note that this value is only used if the font specified by the combination **FontSource** and **FontNumber** is scalable and proportional.

**FontItalic** boolean If true, an italic or oblique font is requested.

**FontNumber** integer This parameter selects the font within the current **FontSource**. Applicable values are determined based upon the **FontSource** and the number of fonts that are available from that font source. Use font numbers found printed on the font sample page. If a **FontNumber** is specified which is outside the range, the value 0 is used instead.

**FontPitch** integer This is the number of characters-per-inch for mono-space scalable fonts. The value is multiplied by 100 to avoid floating-point representations. Thus, to select a 12-pitch font, use the value 1200. This parameter is only used by the PCL5 interpreter if the font specified by the combination **FontSource** and **FontNumber** is scalable and mono-space.

**FontSource** integer This parameter selects the source of the desired font. Current values are: 0 = Internal font, 1 = Downloaded font. Additionally, the value -1 is used when selection of default font is not to be done. If the -1 value is used, then the default font is selected via an obsolete method which uses the parameters **FontFixed**, **FontItalic**, **FontWeight** and **FontTypeface**. If it is not -1, these four parameters are not used to select the font.

**FontSymbolSet** integer This value is the equivalent of the Symbol Set code. The applicable values are described in Hewlett-Packard manuals. Note that this value is only consulted if the font specified by the combination **FontSource** and **FontNumber** is an unbound font. There are 35 legal values.

4, /* OD “ISO-60 Norweg” */
6, /* OF “ISO-25 French” */
7, /* OG “German” */
9, /* OI “ISO-15 Italian” */
11, /* OK “ISO-14 JISASCII” */
14, /* ON “ECMA-94 Latin 1” */
19, /* OS “ISO-11 Swedish” */
21, /* OU “ISO-6 ASCII” */
36, /* 1D “ISO-61 Norweg” */
37, /* 1E “ISO-4 UK” */
FontTypeface integer  This parameter describes the typeface (for example, Times, Helvetica, Palatino, etc.). The integer value (which can be up to 16 bits) comes from a table published by Hewlett-Packard.

FontWeight integer  This value, between -7 and +7, describes the “weight” or “boldness” of the font. -7 is very light and +7 is very bold.

Landscape boolean  If true, the default orientation of the page is landscape unless otherwise specified in the PCL of the page.

LineWrap boolean  If true, long lines wrap to the next line. If false, long lines are truncated.

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

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

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

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

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

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

**Type** name  
**Type** has the value of **Emulator**. For the general definition of **Type**, see Table 3.3 on page 51.

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

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

---

**Table 3.31 Parameters present for the LaserJetIII emulator**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copies</td>
<td>integer</td>
<td>This parameter specifies the default number of copies of a document to be printed.</td>
</tr>
<tr>
<td>FontFixed</td>
<td>boolean</td>
<td>If true, a fixed pitch font is requested. If it is false, a proportional spaced font is requested.</td>
</tr>
<tr>
<td>FontHeight</td>
<td>real</td>
<td>This parameter specifies the desired font height in 1/72 of an inch units.</td>
</tr>
<tr>
<td>FontItalic</td>
<td>boolean</td>
<td>If true, an italic (or oblique) font is requested.</td>
</tr>
<tr>
<td>FontPitch</td>
<td>real</td>
<td>This parameter is used only if <strong>FontFixed</strong> is true. <strong>FontPitch</strong> takes a real number specifying the number of characters per inch.</td>
</tr>
</tbody>
</table>
FontSymbolSet
type: integer
This parameter specifies the mapping from 7 or 8 bit numbers to glyphs that appear on the page. The value of this parameter is the number associated with this field in a downloaded font.

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

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

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

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

ManualFeed
* boolean

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

Type
* name
Type has the value of Emulator. For the general definition of Type, see Table 3.3 on page 51.

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 3.32 Parameters present 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>ColorSetup</td>
<td>string</td>
<td>This parameter allows the user to change the default pen color. The ColorSetup parameter is a string which contains a list of numbers. There must be a multiple of five numbers in the string. Each set of five specifies the pen number (integer), width of the pen’s line in millimeters (real), the red color value (real, between 0 and 1.0), the green color value (real, between 0 and 1.0), and the blue color value (real, between 0 and 1.0).</td>
</tr>
<tr>
<td>Key</td>
<td>Type</td>
<td>Semantics</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AutoLF</td>
<td>boolean</td>
<td>If true, automatic line feeding is specified.</td>
</tr>
<tr>
<td>BoldFontName</td>
<td>name</td>
<td>This parameter specifies the name of the PostScript language font used for boldface printing when ECS is false.</td>
</tr>
<tr>
<td>ECS</td>
<td>boolean</td>
<td>If true, the printer emulates the IBM PC Graphics ECS (extended character set) print wheel. If false, the printer emulates the 96-character plastic print wheel.</td>
</tr>
<tr>
<td>ECSDataWidth</td>
<td>integer</td>
<td>Selects 7- or 8-bit data when ECS is true.</td>
</tr>
<tr>
<td>Pitch</td>
<td>integer</td>
<td>The font width and initial HMI is determined from Pitch. Pitch can have a value of 10, 12 or 15. Any other value will result in a rangecheck error.</td>
</tr>
<tr>
<td>RegFontName</td>
<td>name</td>
<td>This parameter specifies the name of the PostScript language font used for regular printing when ECS is false.</td>
</tr>
<tr>
<td>Type</td>
<td>name</td>
<td>Type has the value of Emulator. For the general definition of Type, see Table 3.3 on page 51.</td>
</tr>
</tbody>
</table>
3.6 The Fax Environment Interface

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

3.6.1 Fax Device Parameters

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

3.6.2 The %Fax% Device

The %Fax% device is used to control the general, global operation of the fax capability of the fax printer. The semantics of the various parameters in the %Fax% device parameter set are described in Table 3.34.

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivityReport</td>
<td>boolean</td>
<td>If true, then an activity report is printed automatically whenever the activity buffer is full. If false, then no activity reports are printed automatically. Printing of the report does not clear the recorded information; the oldest entries are overwritten with new entries. Reports may always be generated by request from the host computer (see also section 3.6.4, “Administrative Resources.”). The default is false.</td>
</tr>
<tr>
<td>DefaultCaptionOn</td>
<td>boolean</td>
<td>This parameter determines whether the default page caption routine will place captions on the fax pages. The default value is true.</td>
</tr>
</tbody>
</table>

Legal values: true, false

Errors: typecheck
DefaultConfirmOn
  boolean This parameter determines whether confirmation reports are produced by the
default procedures. The default value is true.

  Legal values: true, false
  Errors: typecheck

DefaultCoversOn
  boolean This parameter determines whether the default page cover procedure should
actually generate cover pages. The default value is true.

  Legal values: true, false
  Errors: typecheck

DefaultResolution
  integer If the value of FaxType in the FaxOptions dictionary is null, then the value
of this parameter determines the resolution of the raster fax being prepared.
It must have a value of 0 or 1; 1 is the default and specifies to use fine
CCITT group 3 resolution.

  Legal values: 0, 1
  Errors: rangecheck, typecheck

DefaultRetryCount
  integer This integer parameter is the number of retries attempted if the value of
MaxRetries in the options dictionary is null. It must be in the range of 0 to
100 inclusive. The default value is 0.

  Legal values: Any non-negative integer in the range of 0 to 100 inclusive.
  Errors: rangecheck, typecheck

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

  Legal values: Any positive integer in the range of 1 to 60.
  Errors: rangecheck, typecheck
DialToneWaitPeriod

integer

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

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

Errors: rangecheck, typecheck

Group3Adjustment

integer

This parameter determines how received raster faxes are processed for printing.

A value of 0 causes the received raster to be upsampled 2-to-3 (for fine mode). If the resulting image is too long to fit on one page of the selected medium, a continuation page will be printed containing the remainder of the page, assuming it is at least 1/10 inch long. The medium is selected using PageSize Policy 5.

A value of 1 invokes a method which treats the raster as if it were 300x300 dpi (for fine raster data). The net effect of this is that pages are printed at approximately two thirds original size with no continuation pages—even for legal on letter. The default medium for the FaxReceived device is used.

A value of 2 causes the raster to be scaled for best fit on a single page which is selected using PageSize Policy 5. Halftones may be severely marred by this technique.

Values of 10001 and above are used to invoke a user-programmable procedure. The procedure ProgrammableGroup3Adjustments contained in the writeable ProcSet instance FaxDefaultProcs is called. When this procedure is called, a dictionary on the dictionary stack contains the following entries describing the raster needing to be printed:

<table>
<thead>
<tr>
<th>Key</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstNonblank</td>
<td>Integer line number of first non-blank line in the raster data.</td>
</tr>
<tr>
<td>lastNonBlank</td>
<td>Integer line number of last non-blank line.</td>
</tr>
<tr>
<td>nRows</td>
<td>Integer number of scan lines in the raster data.</td>
</tr>
<tr>
<td>highres</td>
<td>Boolean saying whether the raster data is in fine mode or not.</td>
</tr>
<tr>
<td>strm</td>
<td>A file object which may be read to obtain the raster data.</td>
</tr>
</tbody>
</table>

Legal values: Any positive integer in the range of 0 to 20000.

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

Legal values: A string of up to 20 characters.

Errors: limitcheck, typecheck

LocalLanguage string This parameter provides the name of the natural language to use when printing transmission reports and activity logs. The default is (English). The other values supported by all version 2014 (or greater) products with fax support are (Dutch), (French), (German), (Italian) and (Spanish). Translation dictionaries for other languages may also be loaded into the printer. When a report is to be generated, if a translation dictionary for the named language is not present, the one for English will be used. For more information on translation dictionaries see section 3.6.5.

Legal values: A string of up to 50 characters.

Errors: limitcheck, typecheck

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

Legal values: Product dependent

Errors: rangecheck, typecheck

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

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

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

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

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

**Errors:** invalidaccess, limitcheck, typecheck

**ProtocolVersion** string (Read only) This parameter reports the version number of the T.30 fax code present in the printer.

**Legal values:** string

**Errors:** none

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

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

**Errors:** typecheck

**Rings** integer In order to allow a telephone to be shared by the fax printer and a person, the printer needs to let the phone ring several times to give the person time to answer. If the phone is not shared, it is desirable for the printer to answer the phone as quickly as possible. The integer value supplied with this key determines how many rings of the telephone are to be ignored before the fax
printer answers. For example if the value is 2 the printer will attempt to answer the phone just after the second ring. The minimum value is 1, the maximum 10.

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

**Errors:** rangecheck, typecheck

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

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

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

**Legal values:** 0, 1, 2, or 3

**Errors:** rangecheck, typecheck

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

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

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

**Legal values:** 0, 1, or 2

**Errors:** rangecheck, typecheck

**StorageDevice** string  This parameter designates the name of the storage device used to hold received fax data (either PostScript language files or raster) before it is printed, and to hold outgoing fax data (either PostScript language files or raster) before it is transmitted. The default is a value of (%ram%) which means that printer RAM is to be used. For devices other than %ram% (such as a disk or cartridge), a parameter set must exist and the parameters Mounted and Writeable must be defined and set to true.
If there is a disk on the fax printer, setting this parameter to its name (for example, `(%disk0%)`) will mean that the disk is used for storage. Because there is typically more space available on disk than in RAM, this will mean that longer faxes can be received and that longer faxes can be transmitted with just one phone call. Changes to this parameter do not take effect until the fax printer is re-initialized.

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

**Errors:** ioerror, limitcheck

**Type**

(Read only) This parameter always has a value of `/Parameters`. For the general definition of Type, see Table 3.3 on page 51.

**Legal value:** `/Parameters`

**Errors:** none

**WaitForDialTone**

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

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

**Errors:** typecheck

3.6.3 The %Calendar% Device

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

The string (`%Calendar%`) identifies the calendar device and the entries in the dictionary describe the local date and time. The entries in the dictionary are described in Table 3.35.
### Table 3.35 Parameters present in the %Calendar% parameter set

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day</td>
<td>integer</td>
<td>This parameter represents the day.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> An integer in the range of 1 to 31.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> rangecheck</td>
</tr>
<tr>
<td>Hour</td>
<td>integer</td>
<td>This parameter represents the hour.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> An integer in the range of 0 to 23.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> rangecheck</td>
</tr>
<tr>
<td>Minute</td>
<td>integer</td>
<td>This parameter represents the minute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> An integer in the range of 0 to 59.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> rangecheck</td>
</tr>
<tr>
<td>Month</td>
<td>integer</td>
<td>This parameter represents the month.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> An integer in the range of 1 to 12.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> rangecheck</td>
</tr>
<tr>
<td>Running</td>
<td>boolean</td>
<td>This parameter turns the clock off and on. When turning the clock on (setting the value to true), the time elements should also be set at the same time in order to avoid a rangecheck error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The clock must be on in order to set the time. If the clock is turned off (to preserve battery power) or is assumed to be inaccurate, the time returned is January 1, 1980 00:00:00.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> true, false</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> rangecheck</td>
</tr>
<tr>
<td>Second</td>
<td>integer</td>
<td>This parameter represents the second.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Legal values:</strong> An integer in the range 0 to 59.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Errors:</strong> rangecheck</td>
</tr>
</tbody>
</table>
Year  integer  This parameter represents the year. The value of this parameter returned by \texttt{currentdevparams}, has special significance. If it is non-zero and in the range 1980 to 2079, then it represents the year. If it is 0, then the clock is either turned off (to preserve battery power) or is assumed to be inaccurate.

\textbf{Legal values:} An integer in the range 1980 to 2079.

\textbf{Errors:} rangecheck

### 3.6.4 Administrative Resources

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

```
/FaxAdminOps /ProcSet findresource begin
```

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

**Job Records**

The fax printer keeps logs on various aspects of all transmitted and received fax jobs. These records can be accessed by using the four operators described in Table 3.36.

**Table 3.36 Operators for accessing fax job logs**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{proc deletejobsforall}</td>
<td>Selectively deletes entries from the logs.</td>
</tr>
<tr>
<td>\texttt{proc jobsforall}</td>
<td>Allows programmers to perform other functions besides reports.</td>
</tr>
<tr>
<td>\texttt{proc reportjoblist}</td>
<td>Prints one or more pages on the printer listing the log entries. This is similar to the activity report which may be generated automatically. For more information, see the %Fax% parameter \texttt{ActivityReport} in Table 3.34. See also section 2.2.5.</td>
</tr>
<tr>
<td>\texttt{proc returnjoblist}</td>
<td>Sends an ASCII string to the connected computer made up of individual log entries. The components of each log entry are separated by double colons (::).</td>
</tr>
</tbody>
</table>
Each of these operators goes through the list of log entries. For each entry, a job dictionary is placed on the operand stack, and then the procedure proc is executed. proc is expected to consume the job dictionary from the operand stack. For deletejobsforall, the proc must leave a boolean on the stack indicating whether the log entry should be deleted—true means delete. For reportjoblist and returnjoblist, the proc must leave a boolean on the stack indicating whether the entry is to be included in the report—true means it is included. For jobsforall the proc should carry out whatever actions it desires and not push a return value on the stack.

Table 3.37 Entries in a job dictionary

<table>
<thead>
<tr>
<th>Entry</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallCount</td>
<td>integer</td>
<td>The value of CallCount indicates the number of calls involved in this transmission; it is also the number of entries in the Calls array. This value is always 1 for received faxes.</td>
</tr>
<tr>
<td>Calls</td>
<td>array of dictionaries</td>
<td>The dictionaries in the Calls array describe the individual calls of the possibly multiple call fax job. The contents of each dictionary in Calls are described in Table 3.38.</td>
</tr>
<tr>
<td>DialCallee</td>
<td>string</td>
<td>See the description of DialCallee in Table 2.3.</td>
</tr>
<tr>
<td>ErrorArray</td>
<td>array of strings</td>
<td>This is an array of strings which describe status conditions. It is indexed by the ErrorIndex entries described in Table 3.38.</td>
</tr>
<tr>
<td>JobId</td>
<td>integer</td>
<td>This entry is a unique identifier for this send or receive job.</td>
</tr>
<tr>
<td>RecipientName</td>
<td>string</td>
<td>For transmitted faxes, the value of RecipientName is the same as was given in the job’s FaxOptions dictionary. If RecipientName in the job’s FaxOptions dictionary is null, a non-null value to store in the log is sought according to the scheme described under RecipientName in Table 2.3. For received faxes, this item is not present.</td>
</tr>
<tr>
<td>RecipientOrg</td>
<td>string</td>
<td>For transmitted faxes, the value of RecipientOrg is the same as was given in the job’s FaxOptions dictionary. If RecipientOrg in the job’s FaxOptions dictionary is null, a non-null value to store in the log is sought according to the scheme described under RecipientOrg in Table 2.3. For received faxes, this item is not present.</td>
</tr>
<tr>
<td>RecipientPhone</td>
<td>string</td>
<td>For transmitted faxes, the value of RecipientPhone is the same as was given in the job’s FaxOptions dictionary. If RecipientPhone in the job’s FaxOptions dictionary is null, a non-null value to store in the log is sought according to the scheme described under RecipientPhone in Table 2.3.</td>
</tr>
</tbody>
</table>
For received faxes, this item is not present.

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

For received faxes, this item is not present.

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

For received faxes, this item is not present.

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

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

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

Table 3.38 describes the entries in a Calls dictionary.

**Table 3.38  Entries in a Calls dictionary**

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>group 3 raster transmission</td>
</tr>
<tr>
<td>1</td>
<td>PostScript language file transmission</td>
</tr>
<tr>
<td>2</td>
<td>group 3 raster reception</td>
</tr>
<tr>
<td>3</td>
<td>PostScript language file reception</td>
</tr>
</tbody>
</table>
3.6 The Fax Environment Interface

<table>
<thead>
<tr>
<th>Pages</th>
<th>integer</th>
<th>This key designates the number of pages prepared for this call or received. Compare with the definition of PagesSent.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PagesSent</td>
<td>integer</td>
<td>This key designates the number of pages actually transmitted or printed. Compare with the definition of Pages.</td>
</tr>
<tr>
<td>TimeSent</td>
<td>array of integers</td>
<td>This key designates the time when the call started.</td>
</tr>
</tbody>
</table>

**tryphone**

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

```
string int tryphone string
```

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

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

The tryphone operator will return an I/O error if fax transmission is not enabled (see ServiceEnable in Table 3.34) or if the fax hardware is not properly installed.

**Job Queues**

Two operators in the FaxAdminOps ProcSet instance may be used to examine the status of incoming and outgoing faxes which are currently in process or have recently finished being sent or received.

**Table 3.39 Operators for accessing job queues**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proc transmitjobsforall</td>
<td>This operator goes through all transmit jobs that the fax printer is aware of. For each job it composes a dictionary describing the state of that job, pushes the dictionary on the stack and calls the proc. The proc is free to do whatever it wishes, but must end up removing the dictionary from the stack. In the</td>
</tr>
</tbody>
</table>
group of jobs that the operator enumerates will be
a) all finished transmit jobs which still have records in the non-volatile log
storage,
b) jobs waiting to be transmitted because they have asked for delayed trans-
mission, are waiting to retry, or simply have not had their turn yet on the
phone,
c) the job currently using the modem if it is a transmit job.

Refer to Table 3.40 for details on the dictionary created by
transmitjobsforall.

***proc receivejobsforall***

This operator goes through all received jobs that the fax printer is aware of.
For each job it composes a dictionary describing the state of that job, pushes
the dictionary on the stack and calls the proc. The proc is free to do whatever
it wishes, but must end up removing the dictionary from the stack. In the
group of jobs that the operator enumerates will be
a) all finished receive jobs which still have records in the non-volatile log
storage,
b) jobs waiting to be printed,
c) the job currently using the modem if it is a receive job.

Refer to Table 3.41 for details on the dictionary created by
receivejobsforall.

---

**Table 3.40 transmitjobsforall generated dictionary**

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>HostJobID</td>
<td>integer</td>
<td>from job’s options dictionary</td>
</tr>
<tr>
<td>Status</td>
<td>integer</td>
<td>the following integers map to these meanings:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Code</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>DialCallee</td>
<td>string</td>
<td>the dial string used to dial out</td>
</tr>
<tr>
<td>Duration</td>
<td>integer</td>
<td>total number of seconds on the phone spent by this job so far</td>
</tr>
<tr>
<td>ErrorIndex</td>
<td>integer</td>
<td>summary error code; index into ErrorArray</td>
</tr>
<tr>
<td>ErrorArray</td>
<td>array of strings</td>
<td>array of strings describing status conditions</td>
</tr>
<tr>
<td>Kind</td>
<td>integer</td>
<td>0 for raster fax, 1 for PostScript language file transmission jobs (i.e., sending the PostScript language file)</td>
</tr>
</tbody>
</table>
Pages integer number of pages (excluding covers) sent if raster and job is finished or number prepared if job is not finished; 1 for PostScript file transmission jobs

Time array of integers This array has the same format and range of values as the MailingTime array. For finished jobs (Status 0 or 1) this is the time when the (last) call started. For jobs waiting to be sent, this is when the job is scheduled to be sent; this also covers retries. For jobs that are currently being sent this is the current time.

RetriesTried integer number of unsuccessful calls; meaningful only for waiting or sending jobs

RetriesLeft integer number of retries left for this job; meaningful only for waiting or sending jobs

---

**Table 3.41** receivejobsforall generated dictionary

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>integer</td>
<td>the following integers map to these meanings:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>CallerID</td>
<td>string</td>
<td>station ID of caller</td>
</tr>
<tr>
<td>Duration</td>
<td>integer</td>
<td>total number of seconds on the phone spent by this job so far</td>
</tr>
<tr>
<td>ErrorIndex</td>
<td>integer</td>
<td>summary error code; index into ErrorArray</td>
</tr>
<tr>
<td>ErrorArray</td>
<td>array of strings</td>
<td>array of strings describing status conditions</td>
</tr>
<tr>
<td>Kind</td>
<td>integer</td>
<td>0 for raster fax, 1 for PostScript language file transmission jobs</td>
</tr>
<tr>
<td>Pages</td>
<td>integer</td>
<td>number of pages received if raster, 1 if PostScript file transmission jobs</td>
</tr>
<tr>
<td>Recipient</td>
<td>string or null</td>
<td>subaddress field or other in-bound routing info</td>
</tr>
<tr>
<td>Time</td>
<td>array of integers</td>
<td>time the call started</td>
</tr>
</tbody>
</table>

---

**3.6.5 Translations Dictionaries in the FaxDefaultProcs ProcSet**

As noted earlier, a writeable ProcSet instance called FaxDefaultProcs contains the default procedures for cover sheets, transmission reports, page captions and activity reports. These procedures all generate their text by
looking in a language specific dictionary of messages. The dictionary to use is selected by the procedure from a dictionary of such dictionaries based on a language key. This dictionary of translation dictionaries is contained in the FaxDefaultProcs ProcSet instance and can therefore be overwritten. Thus, the text of existing messages can be changed, and entirely new languages can be added. Six translation dictionaries are present initially—those for English, German, French, Spanish, Italian and Dutch. The fonts used with a particular language are determined by entries in that language’s translation dictionary, and are therefore also changeable.

The dictionary of translation dictionaries located in the FaxDefaultProcs ProcSet instance is called TranslationDicts:

```
/TranslationDicts <<
  /languageName1 translationDict1
  /languageName2 translationDict2
  ...
  /languageNameN translationDictN
>> bind def
```

Each translationDict contains the same set of keys. These are described below in Table 3.42.

Many keys are attached to string. Some of the strings are used to generate messages containing variable information such as the number of pages or number of phone calls. The characters “%1” (and sometimes also “%2” and “%3”) are used to indicate the points at which the variable data will be inserted. For example, the translation dictionary entry

```
(We sent %1 pages.)
```

can be used to generate messages on a report such as

```
We sent 15 pages.
```

Similarly,

```
(It took %1 calls, lasting a total of %2 seconds.)
```

could give rise to

```
It took 3 calls, lasting a total of 341 seconds.
```

Other keys in the translationDict are attached to procedures which return a font on the stack. These procedures are called by the default report procedures to obtain the correct fonts for various uses—page titles, column heading, variable data, and so on. The report procedures scale the returned fonts to the sizes needed. The fonts which are returned are assumed to have IOSLatin1Encoding.
## Table 3.42 Entries in a translationDict

<table>
<thead>
<tr>
<th>Entry</th>
<th>Default value in the English dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Items used on more than one report</strong></td>
<td></td>
</tr>
<tr>
<td>/ToLabel</td>
<td>(To:)</td>
</tr>
<tr>
<td>/FromLabel</td>
<td>(From:)</td>
</tr>
<tr>
<td>/PagesLabel</td>
<td>(Pages)</td>
</tr>
<tr>
<td>/FaxNumberLabel</td>
<td>(Fax Number)</td>
</tr>
<tr>
<td>/PageLabel</td>
<td>(Page)</td>
</tr>
<tr>
<td>/NameLabel</td>
<td>(Name)</td>
</tr>
<tr>
<td>/ExcludingCovers</td>
<td>(%1, excluding cover sheets.)</td>
</tr>
<tr>
<td><strong>Items used by the cover page procedure</strong></td>
<td></td>
</tr>
<tr>
<td>/CoverSheetTitle</td>
<td>(Facsimile Cover Sheet)</td>
</tr>
<tr>
<td>/BackCoverTitle</td>
<td>(Facsimile Back Cover)</td>
</tr>
<tr>
<td>/FollowingPages</td>
<td>(Please forward the following pages to:)</td>
</tr>
<tr>
<td>/PrecedingPages</td>
<td>(Please forward the preceding pages to:)</td>
</tr>
<tr>
<td>/RecipientLabel</td>
<td>(Recipient)</td>
</tr>
<tr>
<td>/OrganizationLabel</td>
<td>(Organization)</td>
</tr>
<tr>
<td>/PhoneNumberLabel</td>
<td>(Phone Number)</td>
</tr>
<tr>
<td>/MailStopLabel</td>
<td>(Mail Stop)</td>
</tr>
<tr>
<td>/SenderLabel</td>
<td>(Sender)</td>
</tr>
<tr>
<td>/DateLabel</td>
<td>(Date)</td>
</tr>
<tr>
<td>/SubjectLabel</td>
<td>(Subject)</td>
</tr>
<tr>
<td>/NoteLabel</td>
<td>(Note)</td>
</tr>
<tr>
<td>/WhomIt</td>
<td>(Whom it may concern)</td>
</tr>
<tr>
<td>/Unspecified</td>
<td>(Unspecified)</td>
</tr>
<tr>
<td>/PSTransDesc</td>
<td>(--- PostScript language Transmission ---)</td>
</tr>
</tbody>
</table>
/UnknownDueTo (PostScript language Transmission)

*Items use by the confirmation report procedure*

/TransRepHeader (Transmission Status Report)

/Successful (The following transmission completed successfully.)

/NotSuccessful (The following transmission was not successful.)

/LocationLabel (Location:)

/CallerIDLabel (Caller ID:)

/CalleeIDLabel (Callee ID:)

/PrinterNameLabel (Printer name:)

/PagesUnknownPS (PostScript language transmission.)

/MailedLabel (Mailed:)

/StatusLabel (Status:)

/DurationLabel (Duration:)

/SingleCallOneMinAndSecs (1 minute and %1 seconds.)

/SingleCallMinsOneSec (%1 minutes and 1 second.)

/SingleCallOneMinOneSec (1 minute and 1 second.)

/SingleCallMinsSecs (%1 minutes and %2 seconds.)

/SingleCallSecsOnly (%1 seconds.)

/MultiCallOneMinSecs (1 minute and %1 seconds in %3 calls.)

/MultiCallMinsOneSec (%1 minutes and 1 second in %3 calls.)

/MultiCallOneMinOneSec (1 minute and 1 second in %3 calls.)
Items used by the activity report procedure

/ActRepHeader (FAX ACTIVITY REPORT)
/StartTimeLabel (Start Time)
/ActivityLabel (Activity)
/CallDurationLabel (Call Duration)
/PhoneNumberLabel (Phone Number)
/PagesSentOrPrintedLabel (Pages Sent or Printed)
/ExplActCodes (Explanation of Activity codes:)
/TransGroup3 (Transmit group 3)
/TransPS (Transmit PostScript)
/ReceiveGroup3 (Receive group 3)
/ReceivePS (Receive PostScript)
/FaxKindAbbrev [(T)(TPS)(R)(RPS)]

Items used to describe groups of pages

/ExcludingCover (%1, excluding cover sheet.)
/ThisFinalGroup (%1 this final group, %2 entire document.)
/ThisGroupExcludingCovers (%1 this group, excluding cover sheet(s).)
/SeeTrailers (See trailer sheet(s) to follow.)
/AlreadySent (%1 already sent, trailer sheet(s) to follow.)
/NoPagesSent (Fax - No pages sent to destination.)

Error messages

/ErrInRecvPS (Error in received PS code - job aborted)
/OutOfMemForRecv  (out of memory for receive - page incomplete)
/ErrInCovSheetProc (Error in Cover Sheet Procedure)
/ErrInPageCapProc (Error in Page Caption Procedure)
/ErrInConfirmProc (Error in Confirmation Procedure)

/faxerrarray
[ (No problem.)
   (Callee didn’t respond.)
   (High-speed data transmission failed.)
   (Transmission error, disconnected.)
   (Possible success, but imperfect confirmation.
   (Callee rejected PostScript language transmission.)
   (Error in PostScript language file.)
   (No dial tone.)
   (Internal PS fax error.)
   (Callee wasn’t a PostScript language server.)
   (There were no pages to send.)
   (Storage to hold incoming data was exhausted.)
   (Storage to assemble PostScript language Transmission exhausted.)
   (PS transmission rejected; Error during revert.)
   (Callee number busy)
   (Callee number blacklisted)
]

_items used to generate dates

/sweekdays     [(Sun)(Mon)(Tue)(Wed)(Thu)(Fri)(Sat)(Sun)]
/InvalidTime   (<<Invalid time>>)
/CaptionFont procedure returning Courier with ISOLatin1Encoding
/CoverLabelFont procedure returning Helvetica-Bold with ISOLatin1Encoding
/CoverDataFont procedure returning Bookman-Light with ISOLatin1Encoding
In Level 2, PostScript language objects such as fonts, patterns, filters, and so on, can be managed as open-ended collections of resources grouped into categories. A resource is requested by giving the resource category and name. If the resource does not reside in VM, the resource management mechanism loads it from an external source, such as a disk, a ROM cartridge, or a network file server. Named resources are discussed in section 3.9, “Named Resources” in the PostScript Language Reference Manual, Second Edition.

The resources listed in Tables 4.1, 4.2, and 4.3 are typically present in all implementations of PostScript language version 2014.

4.1 Regular Resources

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

Table 4.1 Regular resources

<table>
<thead>
<tr>
<th>Category name</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font</td>
<td>Product Specific. A product with 35 fonts might list:</td>
</tr>
<tr>
<td></td>
<td>AvantGarde-Book</td>
</tr>
<tr>
<td></td>
<td>AvantGarde-BookOblique</td>
</tr>
<tr>
<td></td>
<td>AvantGarde-Demi</td>
</tr>
<tr>
<td></td>
<td>AvantGarde-DemiOblique</td>
</tr>
<tr>
<td></td>
<td>Bookman-Demi</td>
</tr>
<tr>
<td></td>
<td>Bookman-DemilOblique</td>
</tr>
<tr>
<td></td>
<td>Bookman-Light</td>
</tr>
<tr>
<td></td>
<td>Bookman-LightItalic</td>
</tr>
<tr>
<td></td>
<td>Courier</td>
</tr>
<tr>
<td></td>
<td>Courier-Bold</td>
</tr>
<tr>
<td></td>
<td>Courier-BoldOblique</td>
</tr>
<tr>
<td></td>
<td>Courier-Oblique</td>
</tr>
<tr>
<td></td>
<td>Helvetica</td>
</tr>
<tr>
<td></td>
<td>Helvetica-Bold</td>
</tr>
<tr>
<td></td>
<td>Helvetica-BoldOblique</td>
</tr>
<tr>
<td></td>
<td>Helvetica-Narrow</td>
</tr>
<tr>
<td></td>
<td>Helvetica-Narrow-Bold</td>
</tr>
<tr>
<td></td>
<td>Helvetica-Narrow-BoldOblique</td>
</tr>
</tbody>
</table>
4.2 Implicit Resources

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

Table 4.2 Resources whose instances are implicit

<table>
<thead>
<tr>
<th>Category name</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td>ASCII85Decode ASCII85Encode</td>
</tr>
<tr>
<td></td>
<td>ASCIIHexDecode ASCIIHexEncode</td>
</tr>
<tr>
<td></td>
<td>CCITTFaxDecode CCITTFaxEncode</td>
</tr>
<tr>
<td></td>
<td>DCTDecode DCTEncode</td>
</tr>
</tbody>
</table>

Encoding ISOLatin1Encoding StandardEncoding
Form No instances defined.
Pattern No instances defined.
ProcSet Test
StartPage {see section 2.2.3}
FaxOps {see section 2.2.3}
FaxAdminOps {see section 3.6.4}
FaxDefaultProcs {see sections 2.2.5 and 3.6.5}
ColorSpace No instances defined.
Halftone DefaultHalftone
ColorRendering DefaultColorRendering
OutputDevice Default Printer
Fax FaxReceived {see section 4.4}
HWOptions {see section 4.5}
<table>
<thead>
<tr>
<th>ColorSpaceFamily</th>
<th>CIEBasedA</th>
<th>CIEBasedABC</th>
<th>DeviceCMYK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DeviceGray</td>
<td>DeviceRGB</td>
<td>Indexed</td>
</tr>
<tr>
<td></td>
<td>Pattern</td>
<td>Separation</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Emulator</th>
<th>LaserJetIII</th>
<th>LaserJetIIIP</th>
<th>HP7475A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Diablo630</td>
<td>EpsonFX850</td>
<td>ProprinterXL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IDevice</th>
<th>%Serial%</th>
<th>%Serial_NV%</th>
<th>%Serial_Pending%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>%Parallel%</td>
<td>%Parallel_NV%</td>
<td>%Parallel_Pending%</td>
</tr>
<tr>
<td></td>
<td>%ScsiComm%</td>
<td>%ScsiComm_NV%</td>
<td>%ScsiComm_Pending%</td>
</tr>
<tr>
<td></td>
<td>%LocalTalk%</td>
<td>%LocalTalk_NV%</td>
<td>%LocalTalk_Pending%</td>
</tr>
<tr>
<td></td>
<td>%EtherTalk%</td>
<td>%EtherTalk_NV%</td>
<td>%EtherTalk_Pending%</td>
</tr>
<tr>
<td></td>
<td>%LPR%</td>
<td>%LPR_NV%</td>
<td>%LPR_Pending%</td>
</tr>
<tr>
<td></td>
<td>%AppSocket%</td>
<td>%AppSocket_NV%</td>
<td>%AppSocket_Pending%</td>
</tr>
<tr>
<td></td>
<td>%Telnet%</td>
<td>%Telnet_NV%</td>
<td>%Telnet_Pending%</td>
</tr>
<tr>
<td></td>
<td>%RemotePrinter%</td>
<td>%RemotePrinter_NV%</td>
<td>%RemotePrinter_Pending%</td>
</tr>
<tr>
<td></td>
<td>%PrintServer%</td>
<td>%PrintServer_NV%</td>
<td>%PrintServer_Pending%</td>
</tr>
<tr>
<td></td>
<td>%SNMP%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%SysLog%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%TCP%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%UDP%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%IP%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%SPX%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%IPX%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%EthernetPhysical%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%disk0%</td>
<td>%disk1%</td>
<td>%disk2%</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%cartridge%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%rom%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%Engine%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%Scsi%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%Fax%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%Calendar%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%LaserJetIII%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%LaserJetIIIP%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%Diablo630%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>%HP7475A%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ColorRenderingType</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMapType</td>
<td>2, 3, 4, 5, 6, 7, 8</td>
</tr>
<tr>
<td>FontType</td>
<td>0, 1, 3, 4, 5, 6, 42</td>
</tr>
<tr>
<td>FormType</td>
<td>1</td>
</tr>
<tr>
<td>HalftoneType</td>
<td>1, 2, 3, 4, 5, 6</td>
</tr>
<tr>
<td>ImageType</td>
<td>1</td>
</tr>
<tr>
<td>PatternType</td>
<td>1</td>
</tr>
</tbody>
</table>
4.3 Resources Used to Define New Resources

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

**Table 4.3 Resources used in defining new resource categories**

<table>
<thead>
<tr>
<th>Category name</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>ColorRendering</td>
</tr>
<tr>
<td>ColorSpace</td>
<td>ColorSpaceFamily</td>
</tr>
<tr>
<td>Encoding</td>
<td>Filter</td>
</tr>
<tr>
<td>Font</td>
<td>FontType</td>
</tr>
<tr>
<td>FormType</td>
<td>Generic</td>
</tr>
<tr>
<td>HalftoneType</td>
<td>HWOptions</td>
</tr>
<tr>
<td>IODevice</td>
<td>OutputDevice</td>
</tr>
<tr>
<td>PatternType</td>
<td>ProcSet</td>
</tr>
</tbody>
</table>

Generic No instances defined.

4.4 Accessing Product Page Device Capability Information

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

- Enable applications to query product 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 PostScript product; it simply provides a static list of some of the possible capabilities of the product. Over time, Adobe is likely to define new entries in this dictionary to reflect added capabilities. In 2014 products, the entries listed in Table 4.4 are typically present:
Table 4.4 Description of keys present in an instance of the category OutputDevice

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWResolution</td>
<td>Array of HWResolution values which can be supported by the product.</td>
</tr>
<tr>
<td></td>
<td>Each element within the array can be either an array of two numbers</td>
</tr>
<tr>
<td></td>
<td>indicating a discrete HWResolution support or an array of four numbers</td>
</tr>
<tr>
<td></td>
<td>[x1 y1 x2 y2] indicating that the range of HWResolutions between [x1</td>
</tr>
<tr>
<td></td>
<td>y1] and [x2 y2] is supported. Redundant values may be present.</td>
</tr>
<tr>
<td>ManualSize</td>
<td>Array of page sizes for the product which can be fed manually.</td>
</tr>
<tr>
<td></td>
<td>Each element can be either an array of two numbers indicating a discrete</td>
</tr>
<tr>
<td></td>
<td>page size supported or an array of four numbers [x1 y1 x2 y2] indicat-</td>
</tr>
<tr>
<td></td>
<td>ing that the range of page sizes between [x1 y1] and [x2 y2] is sup-</td>
</tr>
<tr>
<td></td>
<td>ported. Redundant values may be present. In a product that does not sup-</td>
</tr>
<tr>
<td></td>
<td>port the ManualFeed page device parameter, the array of page sizes should</td>
</tr>
<tr>
<td></td>
<td>have no entries.</td>
</tr>
<tr>
<td>PageSize</td>
<td>Array of page sizes for the product which can be fed automatically (as-</td>
</tr>
<tr>
<td></td>
<td>suming appropriate media are installed).</td>
</tr>
<tr>
<td></td>
<td>Each element can be either an array of two numbers indicating a discrete</td>
</tr>
<tr>
<td></td>
<td>page size supported or an array of four numbers [x1 y1 x2 y2] indicat-</td>
</tr>
<tr>
<td></td>
<td>ing that the range of page sizes between [x1 y1] and [x2 y2] is sup-</td>
</tr>
<tr>
<td></td>
<td>ported. Redundant values may be present.</td>
</tr>
<tr>
<td>ProcessColorModel</td>
<td>Array of names or strings which indicate the possible colorant models that</td>
</tr>
<tr>
<td></td>
<td>can be chosen on the product.</td>
</tr>
<tr>
<td></td>
<td>An element in the array can be one of the following values: DeviceGray,</td>
</tr>
<tr>
<td></td>
<td>DeviceRGB, DeviceCMYK, DeviceCMY, and DeviceRGBK.</td>
</tr>
</tbody>
</table>

4.5 Accessing Product Hardware Options Information

The resource category HWOptions has been added for the purpose of enumerating those special hardware options that are currently present on a product. The hardware options that are special do not have any other PostScript language facility for indicating that they are present other than this resource category. For example, a given product might have the ability to support Adobe’s PixelBurst™ co-processor. If the co-processor is not currently installed on the product, the HWOptions resource would not list PixelBurst. When the co-processor is installed on a given product, PixelBurst would appear in the HWOptions dictionary. This resource category is optional. Refer to the PostScript Language Addendum for the product you are concerned with for complete details of the hardware options available.
Some possible instances of the **HWOptions** resource category are listed in Table 4.5.

**Table 4.5  Possible instances of the HWOptions resource category**

<table>
<thead>
<tr>
<th>Instance name</th>
<th>Object type</th>
<th>Value(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fax</td>
<td>string</td>
<td>(USModem) (WorldModem)</td>
</tr>
<tr>
<td>Clock</td>
<td>string</td>
<td>(TODClock)</td>
</tr>
<tr>
<td>PixelBurst</td>
<td>string</td>
<td>Version information</td>
</tr>
<tr>
<td>Type1Coprocessor</td>
<td>string</td>
<td>Version information</td>
</tr>
</tbody>
</table>
Since the publishing of the *Postscript Language Reference Manual, Second Edition*, there have been and potentially will continue to be extensions to the Level 2 PostScript language. This chapter lists those extensions for version 2013 and previous versions.

5.1 Type 6 Halftone Dictionary

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

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

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

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

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

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>HalftoneType</td>
<td>integer</td>
<td>(Required) Must be 6.</td>
</tr>
<tr>
<td>Height</td>
<td>integer</td>
<td>(Required) Height of the threshold array, in pixels.</td>
</tr>
<tr>
<td>Thresholds</td>
<td>file</td>
<td>(Required) When <code>sethalftone</code> is used to make a type 6 halftone dictionary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the current dictionary, the next <code>width x height</code> characters are read from</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the file referenced by <code>file</code> and become the current threshold array.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>So <code>file</code> must reference a file open for read or read/write access at the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>time <code>sethalftone</code> is called. The <code>file</code> object can, of course, be the one</td>
</tr>
<tr>
<td></td>
<td></td>
<td>returned by the <code>currentfile</code> operator. In that case, the next `width x</td>
</tr>
<tr>
<td></td>
<td></td>
<td>height` characters are read from the input stream and saved as a threshold</td>
</tr>
<tr>
<td></td>
<td></td>
<td>array. Also, <code>sethalftone</code> closes <code>file</code> if it encounters EOF and leaves it</td>
</tr>
<tr>
<td></td>
<td></td>
<td>open otherwise.</td>
</tr>
<tr>
<td>TransferFunction</td>
<td>procedure</td>
<td>If present, overrides the transfer function specified by <code>settransfer</code> or</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>setcolortransfer</code>. Required in a type 6 halftone dictionary that is used</td>
</tr>
<tr>
<td>Width</td>
<td>integer</td>
<td>(Required) Width of the threshold array, in pixels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2 Type 42 Font Dictionary

PostScript interpreters can now optionally include support for font type 42. The Type 42 font format is a TrueType font with a PostScript language wrapper to make it conform to the PostScript language font model. A printer driver can distinguish a product that supports font type 42 by using the PostScript Printer Description (PPD) file to extract the appropriate query. The query looks at the resource instance `FontType` to see if Type 42 is supported. Refer to the `FontType` instance in Table 4.2 on page 138. For more information about Adobe’s Type 42 Font Format, refer to technical note #5012 titled “The Type 42 Font Format Specification,” Adobe Developer Support.

The TrueType font format was originally developed by Apple Computer, and is currently supported by the Macintosh® and Windows™ 3.1 operating environments. Prior to the PostScript language version 2013, documents containing TrueType fonts could only be sent to PostScript interpreters in one of two ways. For products that support 680X0-class controllers, the TrueType rasterizer could first be downloaded prior to processing the TrueType document. Otherwise the TrueType font would have to be converted into a PostScript language Type 1 or Type 3 font. Both of these methods have disadvantages: the TrueType rasterizer is large and does not work on all platforms, and conversion to Type 1 or Type 3 is not exact.
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 language Level 2 is accomplished in a standard way in the language through the device setup and interpreter parameter operators. However, for compatibility with existing Level 1 PostScript language driver software, which might depend on `statusdict` operators and keys that were often present in Level 1 PostScript products, a collection of `statusdict` operators and keys is included in each Level 2 PostScript language implementation.

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

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

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

Note In the following list, as well as the tables in this chapter, these symbols are used:
† = this compatibility operator is typically present in all 2014 PostScript language implementations.
‡ = this compatibility operator is typically present in all 2014 PostScript language implementations. However, in the absence of the associated feature, it performs no function aside from its documented effect on the operand stack.
ø = this compatibility operator is typically present in all 2014 PostScript imagesetter implementations.
§ = 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:

```
| a3tray†   | a4tray†   | accuratescreensø |
| appletalktype | b5tray§   | buildtime†   |
| byteorder†  | checkpassword†  | checkscreenø |
| defaulttimeout‡  | diskonline | diskstatus |
| dostartpage   | dosysstart  | duplexmode   |
| emulate       | firstside   | hardwareiomialmode‡ |
| initializedisk§ | jobname†  | jobtimeout†   |
| ledgertray‡  | legaltray§  | lettertray‡ |
| manualfeed   | manualfeedtimeout | margins† |
| mirrorprint  | newsheet    | pagecount‡   |
| pagemargin‡  | pageparamsø | pagecount†  |
| printrename† | processcolors | pagewidth† |
| ramsize       | realformat† | pagecount‡  |
| revision†    | sccbatch    | resolution   |
| setaccuratescreensø | setdefaulttimeout‡ § | setinteractive‡ |
| setdostartpage§ | setduplexmode‡ | |
| sethardwareiomialmode‡§ | | |
| setjobtimeout‡ | setmargins§ § | setmirrorprint§ |
| setpage¶ | setpagemargin‘¶ | setpageparams¶ |
| setpagemargin‘¶ | setprintrename§ § | setresolution¶ |
| setpagedevice¶ | setpagemargin‘¶ | |
| setpageparams¶ | setprintrename§ § | |
| setsoftwareiomialmode¶ | setinteractive¶ | |
| settumble‡ | setuserdiskpercent§ | softwareiomialmode‡ |
| tumblem | userdiskpercent | waittimeout‡ |
| 11x17tray§ | | |
```
6.2 Compatibility Operator Descriptions

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

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

checkpassword†
checkscreenø
devforall†
emulate
firstside
newsheet
sccinteractive‡
setpapertray§
setsccinteractive‡§
setuserdiskpercent§

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

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

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

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

6.2.2 Using a Password to Change Persistent Values

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

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

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

Compatibility operators that affect page device parameters save their persistent values only if they are executed from an unencapsulated job. In encapsulated jobs the values set by these compatibility operators will obey the normal save-restore rules and are not saved to persistent storage.
Note  The compatibility objects are present in Level 2 printers for compatibility purposes only and their use in PostScript language Level 2 programs is strongly discouraged.

Table 6.1 statusdict compatibility operators

<table>
<thead>
<tr>
<th>Operator/Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a3tray†</td>
<td>See section 6.2.5, “Paper Tray Operations.”</td>
</tr>
<tr>
<td>a4tray†</td>
<td>See section 6.2.5, “Paper Tray Operations.”</td>
</tr>
<tr>
<td>accuratescreens ‡</td>
<td>See section 6.2.8, “Imagesetter Compatibility Operators.”</td>
</tr>
<tr>
<td>appletalktype</td>
<td>– appletalktype string is a string with the same value as the LocalTalkType device parameter in the %LocalTalk% parameter set and the EtherTalkType parameter in the %EtherTalk% parameter set. Redefining appletalktype will cause the LocalTalkType parameter to change as well as the EtherTalkType parameter. Similarly, changes to the EtherTalkType or the LocalTalkType parameter will change the string returned by the appletalktype operator.</td>
</tr>
<tr>
<td>b5tray†</td>
<td>See section 6.2.5, “Paper Tray Operations.”</td>
</tr>
<tr>
<td>buildtime †</td>
<td>– buildtime int is an integer with the same value as the system parameter BuildTime.</td>
</tr>
<tr>
<td>byteorder †</td>
<td>– byteorder bool is a boolean with the same value as the system parameter ByteOrder.</td>
</tr>
</tbody>
</table>

Errors: stackoverflow
**checkpassword**

```
int checkpassword bool
string checkpassword bool
```

determines whether `string` or `int (int is converted to a string)` is a valid password for either `SystemParamsPassword` or `StartJobPassword`. If `true`, `true` is returned, otherwise `false` is returned. If either password is not set then `true` will be returned. A returned value of `true` indicates that `string` or `int` is a valid argument to `startjob` and `exitserver`. There is no PostScript language Level 2 equivalent for `checkpassword`.

**Errors:** stackunderflow, typecheck

**checkscreen**

See section 6.2.8, “Imagesetter Compatibility Operators.”

**defaulttimeouts**

```
– defaulttimeouts job manualfeed wait
```

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

**Errors:** stackoverflow

**diskonline**

```
– diskonline bool
```

returns `true` if and only if a writeable disk device is mounted. This is determined by searching all device parameter sets named `%disk*%`, where `*` represents zero or more additional characters in the name. If the `Writable` 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 `Writable true` need not have an initialized file system.

**Errors:** stackoverflow

**diskstatus**

```
– diskstatus free total
```

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

**Errors:** stackoverflow
**dostartpage**  

\[ \texttt{dostartpage bool} \]

returns the value of the system parameter \texttt{DoStartPage}.

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

**Errors:**  

\texttt{stackoverflow}

---

**dosysstart**  

\[ \texttt{dosysstart bool} \]

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

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

**Errors:**  

\texttt{stackoverflow}

---

**duplexmode**  

See section 6.2.6, “Page Duplex Compatibility Operators.”

---

**emulate**  

\[ \texttt{input-stream emulation-name emulate} \]

or  

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

Causes the PostScript interpreter to yield control, and the emulator named by \texttt{emulation-name} to start processing. The \texttt{emulate} operator is present in \texttt{statusdict}, and only in products which have one or more emulators co-resident with the PostScript 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 \texttt{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 language context will generally have been lost.

The allowed values of \texttt{emulation-name} may be found in the implicit resource category \texttt{Emulator}. An illegal \texttt{emulation-name} will cause a \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 *input-stream* is a file object which becomes the input source for the emulator. The *input-stream* specified must be appropriate to the product-dependent emulator, as defined in the product *Addendum*. An illegal *input-stream* will cause an *invalidaccess* error.

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

**firstside**

See section 6.2.6, “Page Duplex Compatibility Operators.”

**hardwareiomode ‡**

- hardwareiomode int

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

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

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

**Errors:** stackoverflow

**initializedisk ‡**

`pages action initializedisk` –

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

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

**jobname ‡**

- jobname string

is a string with the same value as the user parameter *JobName*. Redefining either *jobname* or the user parameter *JobName* redefines the other to the same value.
The user parameter **JobName** must be present for the compatibility operator **jobname** to be present.

**Errors:** stackoverflow

**jobtimeout** †

- **jobtimeout** *int*

returns the value of the user parameter **JobTimeout**.

**Errors:** stackoverflow

**ledgertray** †

See section 6.2.5, “Paper Tray Operations.”

**legaltray** †

See section 6.2.5, “Paper Tray Operations.”

**lettertray** ††

See section 6.2.5, “Paper Tray Operations.”

**manualfeed**

- **manualfeed** *bool*

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

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

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

**Errors:** stackoverflow

**manualfeedtimeout**

- **manualfeedtimeout** *int*

is an integer that works in conjunction with the page device parameter **ManualFeedTimeout** to determine the **manualfeed** timeout for any given page. By default, **manualfeedtimeout** is not defined in **statusdict** and in that case the value of the page device parameter **ManualFeedTimeout** is used to determine the timeout value. If a job has defined **manualfeedtimeout** to be an integer value in **statusdict** then this value will be used instead of **ManualFeedTimeout** for the timeout value.
The values of `ManualFeedTimeout` and `manualfeedtimeout` are determined independently. That is, setting the `manualfeedtimeout` integer or setting the `pagedevice` parameter, `ManualFeedTimeout`, does not affect the value of the other.

Errors: stackoverflow

`margins`‡

- `margins top left`

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

Errors: stackoverflow

`mirrorprint`

- `mirrorprint boolean`

Returns the value of the page device parameter `MirrorPrint`.

Errors: none

`newsheet`

See section 6.2.6, “Page Duplex Compatibility Operators.”

`pagecount`‡

- `pagecount int`

returns the value of the system parameter `PageCount`.

Errors: stackoverflow

`pagemargin`ø

See section 6.2.8, “Imagesetter Compatibility Operators.”

`pageparams`ø

See section 6.2.8, “Imagesetter Compatibility Operators.”

`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
\textbf{printername} $^\dagger$ \hfill \textit{string printername substring}

stores the value of the system parameter \texttt{PrinterName} in \textit{string} and returns a string object designating the \textit{substring} actually used.

\textbf{Errors:} rangecheck, stackunderflow, typecheck

\textbf{processcolors} \hfill \textit{processcolors int}

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

\textbf{Errors:} stackoverflow

\textbf{product} $^\dagger$ \hfill \textit{product string}

is a \textit{string} in \texttt{statusdict} initialized to the value of the string \texttt{product} in \texttt{systemdict}.

\textbf{Errors:} stackoverflow

\textbf{ramsize} \hfill \textit{ramsize int}

returns the number of bytes of RAM available to the product. Refer to the \texttt{RamSize} system parameter.

\textbf{Errors:} stackoverflow

\textbf{realformat} $^\dagger$ \hfill \textit{realformat string}

is a \textit{string} with the same value as the system parameter \texttt{RealFormat}.

\textbf{Errors:} stackoverflow

\textbf{resolution} \hfill \textit{resolution bitsperinch}

Returns the first component of the \texttt{HWResolution} array for the current output device.

\textbf{Errors:} stackoverflow
revision† — revision int

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

Errors: stackoverflow

sccbatch

See section 6.2.3, “SCC Operations.”

sccinteractive‡

See section 6.2.3, “SCC Operations.”

setaccuratescreensⁿ

See section 6.2.8, “Imagesetter Compatibility Operators.”

setdefaulttimeouts¹§¶

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

Errors: invalidaccess, rangecheck, stackunderflow, typecheck

setdostartpage§

bool setdostartpage —

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

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

Errors: invalidaccess, stackunderflow, typecheck

setdosysstart§

bool setdosysstart —

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

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

Errors: invalidaccess, stackunderflow, typecheck
**sethardwareiomode**

`int sethardwareiomode`

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

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

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

**setjobtimeout**

`int setjobtimeout` –

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

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

**Errors:** stackunderflow, typecheck

**setmargins**

`top left setmargins` –

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

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

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

*boolean setmirrorprint –*

Creates a new page device with the parameter **MirrorPrint** set to *boolean*.

**Errors:** stackunderflow, typecheck

setpage

See section 6.2.8, “Imagesetter Compatibility Operators.”

setpagemargin

See section 6.2.8, “Imagesetter Compatibility Operators.”

setpageparams

See section 6.2.8, “Imagesetter Compatibility Operators.”

setpagestackorder

*bool setpagestackorder –*

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

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

**Errors:** invalidaccess, stackunderflow, typecheck

setprintername

*string setprintername –*

This compatibility operator 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

setresolution

*bitsperinch setresolution –*

Creates a new page device with the parameter **HWResolution** set to [bitsperinch bitsperinch].

**Errors:** rangecheck, stackunderflow, typecheck

sccbatch

See section 6.2.3, “SCC Operations.”

sccinteractive

See section 6.2.3, “SCC Operations.”
**setsoftwareiomode**

`int setsoftwareiomode` –

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

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

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

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

**settumble**

See section 6.2.6, “Page Duplex Compatibility Operators.”

**setuserdiskpercent**

`int setuserdiskpercent` –

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

**Errors:** `invalidaccess`, `rangecheck`, `stackunderflow`, `typecheck`
softwareiomode ‡ – softwareiomode int

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

Note: If the Interpreter is not one of the values that can be set via setsoftwareiomode, softwareiomode will return -1.

Errors: stackoverflow

See section 6.2.6, “Page Duplex Compatibility Operators.”

tumble

userdiskpercent – userdiskpercent int

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

Errors: stackoverflow

waittimeout ‡ – waittimeout int

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

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

Errors: stackoverflow

See section 6.2.5, “Paper Tray Operations.”

6.2.3 SCC Operations

The SCC (Serial Communications Controller) operators use a byte options argument (an integer parameter with values in the range 0 – 255) that holds an encoding of four SCC parameters: stop bits, data bits, flow control, and parity. The byte is encoded as follows (bit positions 7 – 0 with 7 the high bit and 0 the low bit):
Table 6.2  Stop bits

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

Table 6.3  Data bits

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

Table 6.4  Flow control

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

Table 6.5  Parity

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

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

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

**Table 6.6  Options byte to devparams conversion**

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

**Table 6.7  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 7 bits mark</td>
</tr>
<tr>
<td>7 bits Space 7 bits space</td>
</tr>
<tr>
<td>7 bits Mark 7 bits mark</td>
</tr>
<tr>
<td>7 bits Odd 7 bits odd</td>
</tr>
<tr>
<td>7 bits Even 7 bits even</td>
</tr>
<tr>
<td>8 bits None 8 bits mark</td>
</tr>
<tr>
<td>8 bits Space 8 bits space</td>
</tr>
<tr>
<td>8 bits Mark 8 bits mark</td>
</tr>
<tr>
<td>8 bits Odd 8 bits odd</td>
</tr>
<tr>
<td>8 bits Even 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.

Table 6.8 SCC compatibility operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>sccbatch</td>
<td>channel sccbatch baud options</td>
</tr>
</tbody>
</table>

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

*Note: If the FlowControl parameter is set to /DtrLow, sccbatch will return 1 in bit position 4, 3 and 2. If the FlowControl parameter is set to something other than /XonXoff, /Dtr, /DtrLow, or /EtXAck, sccbatch will return 0 in bit position 4, 3 and 2.*

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

Errors: rangecheck, stackoverflow, stackunderflow, typecheck

sccinteractive ‡

channel sccinteractive baud options

pops the input argument off the stack and pushes 0 0 on the stack. This operator is essentially a no-op.

Errors: invalidaccess, rangecheck, stackoverflow, stackunderflow, typecheck

setsccbatch §

channel baud options setsccbatch –

This compatibility operator sets the communication device parameters for serial communications. Either the %SerialB_NV% (if channel equals 9) or the %Serial_NV% (if channel equals 25) settings are affected. The following device parameters are affected by baud and options: Baud, StopBits, DataBits, FlowControl, Parity, and CheckParity. Baud, StopBits, and FlowControl are set according
to the corresponding values for baud, stop bits and flow control. **DataBits** and **Parity** are set based on Table 6.7 above. **CheckParity** is set according to the new **Parity** setting:

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

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

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

---

**setsccinterative**

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

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

---

### 6.2.4 Paper Size Operations

All the operators in this section are in **userdict**. Each operator executes **setpagedevice** to request a specific paper size. The only difference among these operations is the size of paper requested and the **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 and turns off the normal PostScript language Level 2 media matching mechanism. (For a detailed description of **PageSize Policy** 7 see Table 2.2 on page 14.) The only error that is generated is a **limitcheck** caused by insufficient memory for the requested imaging area. In Table 6.9, default units (1/72 inch) are used as the units for the **PageSize** and **ImagingBBox**.
Table 6.9  Paper size compatibility operators (in userdict)

<table>
<thead>
<tr>
<th>Operator</th>
<th>PageSize</th>
<th>ImagingBBox</th>
</tr>
</thead>
<tbody>
<tr>
<td>letter</td>
<td>[612 792]</td>
<td>null</td>
</tr>
<tr>
<td>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>or [499 709]</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 and lettersmall compatibility operators will be present only if the size [612 792] is an element of the PageSize array in some instance of the OutputDevice resource category.

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

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

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

6.2.5 Paper Tray Operations

All of the operators in this section are in statusdict. Each operator executes setpagedevice to request a tray containing a specific paper size. The only difference among these operations is the size of paper requested. The PageSize requested is the same as for the corresponding page size operator discussed in the previous section and the ImagingBBox requested is always null. These operators use the specified size as indicated below as a page device PageSize parameter.
All of these operators set the **PageSize Policy** to 0, which guarantees that a **configuration error** is generated if a tray containing the requested paper size is not present. The implementation of the compatibility operators convert any such **configuration error** to a **range check** for compatibility with PostScript language Level 1 implementations. Also, a **limit check** error can occur because of insufficient memory for the requested imaging area.

**Table 6.10  Paper tray compatibility operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>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 1191]</td>
<td>null</td>
</tr>
<tr>
<td>a4tray†</td>
<td>[595 842]</td>
<td>null</td>
</tr>
<tr>
<td>b5tray†</td>
<td>[516 729] or [499 709]</td>
<td>null</td>
</tr>
<tr>
<td>11x17tray†</td>
<td>[792 1224]</td>
<td>null</td>
</tr>
</tbody>
</table>

The **lettertray** compatibility operator will be present only if the size [612 792] is an element of the **PageSize** array in some instance of the **OutputDevice** resource category.

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

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

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

### 6.2.6 Page Duplex Compatibility Operators

All compatibility objects described below are defined in **statusdict** unless otherwise specified.
Table 6.11  Page duplex compatibility operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>duplexmode</td>
<td><code>duplexmode bool</code></td>
</tr>
<tr>
<td></td>
<td>returns the value of the page device parameter <code>Duplex</code>.</td>
</tr>
<tr>
<td></td>
<td>The page device parameter <code>Duplex</code> must be present for the compatibility operator <code>duplexmode</code> to be present.</td>
</tr>
<tr>
<td></td>
<td><strong>Errors:</strong> stackoverflow</td>
</tr>
<tr>
<td>firstside</td>
<td><code>firstside bool</code></td>
</tr>
<tr>
<td></td>
<td>returns <code>true</code> if the current page is a front side, <code>false</code> if the current page is a back side.</td>
</tr>
<tr>
<td></td>
<td>Note: This compatibility operator is sometimes found on products that do not support duplex printing. On these products, <code>firstside</code> may be used to generate output that is intended to be copied using a duplex copier.</td>
</tr>
<tr>
<td></td>
<td><strong>Errors:</strong> stackoverflow</td>
</tr>
<tr>
<td>newsheet</td>
<td><code>newsheet</code></td>
</tr>
<tr>
<td></td>
<td>If <code>Duplex</code> is <code>true</code> and the current page is a back-side, causes this page to be printed as is (perhaps blank) and sets up a clean printing environment for the next page. Otherwise, executing newsheet has no effect.</td>
</tr>
<tr>
<td></td>
<td>The page device parameter <code>Duplex</code> must be present for the compatibility operator <code>newsheet</code> to be present.</td>
</tr>
<tr>
<td></td>
<td><strong>Errors:</strong> none</td>
</tr>
<tr>
<td>setduplexmode</td>
<td><code>bool setduplexmode</code></td>
</tr>
<tr>
<td></td>
<td>This compatibility operator sets the page device parameter <code>Duplex</code> to <code>bool</code>.</td>
</tr>
<tr>
<td></td>
<td>The page device parameter <code>Duplex</code> must be present for the compatibility operator <code>setduplexmode</code> to be present.</td>
</tr>
<tr>
<td></td>
<td><strong>Errors:</strong> stackunderflow, typecheck</td>
</tr>
</tbody>
</table>
settumble§

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

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

**Errors:** stackunderflow, typecheck

tumble

returns the value of the page device parameter **Tumble**.

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

**Errors:** stackoverflow

### 6.2.7 Device Compatibility Operators

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

**Table 6.12 Device compatibility operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>devdismount†§</td>
<td>string devdismount –</td>
</tr>
</tbody>
</table>

This compatibility operator sets the device parameter **Mounted** to `false` within 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 effect. Some devices cannot be dismounted. Trying to dismount these will also have no effect.

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

**Errors:** invalidaccess, stackunderflow, undefinedfilename
**devforall** †

`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** †§

`string pages format devformat` -

This compatibility operator sets the **LogicalSize** device parameter of the parameter set corresponding to the device specified by `string` to the value specified by `pages`. It then sets the **InitializeAction**, in the same parameter set, to the value of `format+1`. Refer to the **InitializeAction** and **LogicalSize** file system device parameters for complete details.

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

---

**devmount** †§

`string devmount bool`

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

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

**Errors:** invalidaccess, stackunderflow, undefinedfilename
devstatus †

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.

Attribute Description

searchable The searchable attribute corresponds to the Searchable 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.

writeable The writeable attribute corresponds to the Writeable device parameter and indicates whether files on this device can be written.

hasNames The hasNames attribute corresponds to the HasNames device parameter and is a boolean which indicates whether the device supports named files.

mounted The mounted boolean (Mounted device parameter) indicates whether the device is mounted.

removable The removable boolean (Removable device parameter) indicates whether the media within the device can be removed.

searchOrder The searchOrder attribute (SearchOrder device parameter) indicates the priority at which the device participates when searching for a file in operations in which no device has been specified.

freePages The freePages boolean (Free device parameter) indicates the amount of free space (in pages).

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

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

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

Errors: stackunderflow
6.2.8 Imagesetter Compatibility Operators

Table 6.13 Imagesetter compatibility operators found in statusdict

<table>
<thead>
<tr>
<th>Operator</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>accuratescreens</td>
<td>– accuratescreens boolean</td>
</tr>
<tr>
<td></td>
<td>Returns the value of the user parameter AccurateScreens. A value of true means that accurate screening is enabled.</td>
</tr>
<tr>
<td></td>
<td>Errors: stackoverflow</td>
</tr>
<tr>
<td>checkscreen</td>
<td>freq angle checkscreen actualfreq actualangle moirelength</td>
</tr>
<tr>
<td></td>
<td>Returns the actual screen frequency and angle that would be used if setscreen was called. The moirelength is the distance in inches where the deviation from the requested dot pattern would reach a fixed fraction of a cell size, and is thus a measure of how accurate the actual screen would approximate the requested screen. Note that this operator does not affect the current screen.</td>
</tr>
<tr>
<td></td>
<td>Errors: stackoverflow</td>
</tr>
<tr>
<td>pagemargin</td>
<td>– pagemargin x</td>
</tr>
<tr>
<td></td>
<td>Returns the x value (measured in device units) of the page device parameter PageOffset.</td>
</tr>
<tr>
<td></td>
<td>Errors: stackoverflow</td>
</tr>
<tr>
<td>pageparams</td>
<td>– pageparams width height margin orientation</td>
</tr>
<tr>
<td></td>
<td>Suppose that the value of the page device parameter PageSize is [x y] and that PageOffset is [X Y]. Then pageparams returns values, depending on the value of the page device parameter Orientation, as indicated in the following table:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Orientation</th>
<th>width</th>
<th>height</th>
<th>margin</th>
<th>orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>y</td>
<td>x</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>y</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>2*</td>
<td>y</td>
<td>x</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>3*</td>
<td>x</td>
<td>y</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>
setaccuratescreens

boolean setaccuratescreens

Sets the user parameter AccurateScreens to have the value boolean.

Errors: stackunderflow, invalidaccess, typecheck

setpage

width height orientation setpage

Creates a new page device with the parameter PageSize set to [width height] and Orientation to orientation.

Errors: limitcheck, rangecheck, stackunderflow, typecheck

setpagemargin

margin setpagemargin

Creates a new page device with the parameter PageOffset set to [margin 0].

Errors: rangecheck, stackunderflow, typecheck

setpageparams

width height margin orientation setpageparams

Creates a new page device with the parameter PageSize set to [width height], the x value of PageOffset set to margin, and Orientation set to orientation.

Errors: limitcheck, rangecheck, stackunderflow, typecheck, undefinedresult
Appendix A

Changes Since Earlier Versions

A.1 Changes since Version 2013, March 31, 1993

The following is a brief list of changes made to the PostScript Language Reference Manual Supplement since March 31, 1993. These changes were incorporated into the 2014 release.

- The default value has been added to the HostJobID key on page 23.

- The PageCount parameter on page 47 has an improved description.

- Section 3.5 (page 50) contains a new note to support the removal of the word “typically” from Tables 3.4 to 3.35.

- The description of DelayedOutputClose on page 58 explains better the parameter’s function.

- A description of the Bus key has been added to Table 3.7. See page 68.

- The GatewayAddress parameter has been updated to explain its reliance on the IPAddressDynamic parameter value. See page 89.

- The Removable parameter (page 100) is described in more detail.

- Descriptions of the new page device keys DeferredMediaSelection, ImageShift, and MediaPosition have been added to Table 2.1 found on page 6. Also the InsertSheet and PageOffset descriptions are improved.

- Section 2.1 on page 14 describes better what happens if the Type key for a details dictionary is wrong or missing.

- The DialCallee parameter in the FaxOptions has the comma character better explained (page 22). Also in FaxOptions, there are 2 new keys. They are HostJobID (page 23) and RecipientLanguage (page 25). Also the description for RecipientName and RecipientPhone have been improved.
• Section 2.2.5, “CoverSheet, Confirmation, and PageCaption Procedures” on page 27 has a new improved description.

• Section 2.4.3 explains that undefined is a possible error from setpagedevice.

• Two new imagesetter system parameters have been added to Table 3.2 found on page 41. The new parameters CurBufferType (on page 41) and MinBandBuffers (page 47) are found in this table.

• The semantics of the PageCount system parameter have changed. See page 47.

• The new communication device parameter DelayedOutputClose is described on page 58. This parameter exists in almost all parameter sets of type /Communications.

• The description of the detection of protocol when using AutoSelect has been clarified (page 60).

• The Novell SPX/IPX Node address syntax has been modified since the 2013 Supplement. Refer to Table 3.10 found on page 73.

• The %AppSocket% parameter set has a new parameter called ControlPortNumber (page 76).

• Several new Novell device parameter sets are described. They are %RemotePrinter% (page 80), %PrintServer% (page 81), %SPX% (page 92), and %IPX% (page 93).

• The description for the TrapHost parameter in the %SNMP% set has changed. See page 84.

• The parameter set %UDP% (page 87) has been added to the family of TCP/IP related parameter sets.

• The description of the various parameters in the %IP% set has been rewritten. See Table 3.20 found on page 88.

• The Name parameter in the %EthernetPhysical% set is read-only (page 95).

• The %fax% parameter set has 2 new keys called LocalLanguage (page 119) and Group3Adjustment (page 118).

• Section 3.6.4 has a new subsection called “Job Queues” on page 127 which describes the new capability in 2014 of displaying fax job queues.
• Section 3.6.5, “Translations Dictionaries in the FaxDefaultProcs ProcSet” on page 129 is new.

• Table 4.4, “Description of keys present in an instance of the category OutputDevice” on page 141 has had ProcessColorModel added to it.

• Table 4.5, “Possible instances of the HWOptions resource category” on page 142 lists WorldModem as a possible Fax instance value.

• There are several new instances listed in the IDevice resource category. The list includes %RemotePrinter%, %PrintServer%, %UDP%, %SPX%, and %IPX%. Each has the corresponding “_NV” and “_Pending” set names.

• There are other minor corrections throughout.

A.2 Changes since Version 2012, November 25, 1992

The following is a brief list of changes made to the PostScript Language Reference Manual Supplement since November 25, 1992. These changes were incorporated into the 2013 release.

• The new page device key PageOffset has been added to chapter 2.

• Section 3.5.2 has been re-organized to describe device parameter sets associated with both network and point-to-point communications. Several new parameter sets are described which have been defined to support the use of the TCP/IP protocol over Ethernet. The new parameter sets include %LPR%, %AppSocket%, %Telnet%, %SNMP%, %SysLog%, %TCP%, %IP%, and %EthernetPhysical%.

• In Section 3.5.3 there is a sub-section called “Disk, Cartridge and Rom Parameter Tables” on page 97 which now additionally describes the %rom% device parameter set.

• The LaserJetIII emulator has several additional device parameters that had not previously been documented. Refer to Table 3.30 found on page 110.

• The HP7475A emulator and the Diablo630 emulator device parameter sets both have a Type key which had previously not been documented. Refer to Table 3.32 and Table 3.33 found on page 115. Also the Diablo630 has a Pitch key which had previously not been documented.

• Chapter 4 has been re-written. Some resource instances had previously been missing. Section 4.5 describes the HWOptions resource instance found on some products. The IDevice resource category lists new
instances for %LPR%, %AppSocket%, %Telnet%, %SNMP%, %SysLog%, %TCP%, %IP%, and %EthernetPhysical% which are some new TCP/IP and Ethernet related device parameter sets.

- 2013 PostScript products can optionally contain support for font type 42. The Type 42 font format is a TrueType font with a PostScript wrapper to make it conform to the PostScript font model. TrueType is a font format originally developed by Apple Computer. Refer to section Section 5.2.

- Various compatibility operators typically found on imagesetters and roll fed media devices have been added to chapter 6. The list includes accuratescreens, checkscreen, mirrormap, pagemargin, pageparams, resolution, setaccuratescreens, setpage, setpagemargin, setpageparams, and setresolution.

A.3 Changes since Version 2011, January 24, 1992

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

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

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

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

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

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

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

- There are several new System Parameters introduced in 2012. They are CurStoredFontCache, CurStoredScreenCache, MaxHWRendringBuffer, MaxImageBuffer, MaxStoredFontCache, and MaxStoredScreenCache.
• There are several new Device Parameter sets. The list includes %Scsi-Comm%, %os%, %Scsi%, %Engine%, %LaserJetIII%, %Fax%, and %Calendar%.

• The **Interpreter** key in all of the Device Parameter sets of type /Communications can now be set to LaserJetIII for PCL5 emulation. EpsonFX850 is also a new choice.

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

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

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

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

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

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

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

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

### A
- `a3` 147, 165
- `a3tray` 146, 149, 166
- `a4` 147, 165
- `a4small` 147, 165
- `a4tray` 146, 149, 166
- `a5tray` 146, 149, 166
- `AccurateScreens` 39
- `ActivityReport` 116
- `AdvanceDistance` 5
- `AdvanceMedia` 5
- `appletalktype` 70, 72, 146, 149
- `AutoLF` 115
- `AutoSelect` 60, 61, 65

### B
- `b5` 165
- `Baud` 62, 163
- `BeginPage` 5
- `Binary` 60, 61
- `Bind` 6
- `BindDetails` 6
- `BlockSize` 97, 101
- `BoldFontName` 115
- `Booklet` 6
- `BookletDetails` 6
- `BootDelay` 106
- `BroadcastAddress` 88
- `BSizeStandard` 108
- `BuildTime` 41, 149
- `buildtime` 146, 149
- `Bus` 98
- `ByteOrder` 41, 149
- `byteorder` 146, 149

### C
- `calendar device parameters`
  - `Day` 123
  - `Hour` 123
  - `Minute` 123
  - `Month` 123
  - `Running` 123
  - `Second` 123
  - `Year` 124
- `CallCount` 125
- `CalleeID` 28
- `CalleePhone` 20, 26
- `CallerID` 20, 129
- `CallerPhone` 20
- `CallLength` 28, 126
- `Calls` 125
- `Calls dictionary entries`
  - `CallLength` 126
  - `CoverPagesSent` 126
  - `ErrorArray` 126
  - `ErrorIndex` 126
  - `FaxKind` 126
  - `Pages` 127
  - `PagesSent` 127
  - `TimeSent` 127
- `CartridgeID` 102
- `CartridgeType` 102
- `Category` 44, 140
- `CheckParity` 62, 106, 163, 164
- `checkpassword` 146, 147, 150
- `Checksum` 87, 93
- `Clock` 142
- `Collate` 5
- `ColorRendering` 44, 138
- `ColorRenderingType` 139
- `ColorSetup` 114
- `ColorSpace` 44, 138
- `ColorSpaceFamily` 139
compatibility operators 2
11x17 147, 165
11x17tray 146, 160, 166
a3 147, 165
a3tray 146, 149, 166
a4 147, 165
a4small 147, 165
a4tray 146, 149, 166
accuratescreens 146, 149, 171
appletalktype 70, 72, 146, 149
a4 147, 165
a4small 147, 165
a4tray 146, 149, 166
accuratescreens 146, 149, 171
appletalktype 70, 72, 146, 149

11x17 147, 165
11x17tray 146, 160, 166
a3 147, 165
a3tray 146, 149, 166
a4 147, 165
a4small 147, 165
a4tray 146, 149, 166
accuratescreens 146, 149, 171
appletalktype 70, 72, 146, 149

b5 147, 165
b5tray 146, 149, 166
buildtime 146, 149
byteorder 146, 149
checkpassword 146, 147, 150
checkscreen 146, 147, 150, 171
defaulttimeouts 146, 150
devdismount 147, 168
devforall 147, 169
devformat 147, 169
devmount 147, 149
devstatus 146, 170
diskonline 146, 150
diskstatus 146, 150
dostartpage 146, 151
dosysstart 146, 151
duplexmode 146, 151, 167
eulate 146, 147, 151
exitserver 37, 49, 148, 150
firstside 146, 147, 152, 167
hardwareiomode 146, 152
initializedisk 146, 152
jobname 146, 152, 153
jobtimeout 146, 153
ledger 147, 165
ledgertray 146, 153, 166
legal 147, 165
legaltray 146, 153, 166
letter 147, 165
lettersmall 147, 165
lettertray 146, 153, 166
manualfeed 146, 153
manualfeedtimeout 146, 153
margins 146, 154
mirrorprint 146, 154
newsheet 146, 147, 154, 167
note 147, 165
pagecount 146, 154
pagemargin 146, 154, 171
pageparams 146, 154, 171
pagestackorder 146, 154
printername 146, 155
processcolors 146, 155
product 146, 155
ramsize 146
realformat 146, 155
resolution 146, 155
revision 146, 156
scebatch 146, 156, 158, 163
sccinteractive 146, 147, 156, 158, 163
setaccuratescreens 146, 156, 172
setdefaulttimeouts 146, 156
setdostartpage 146, 156
setdosysstart 146, 156
setduplexmode 146, 167
sethardwareiomode 146, 157
setjobtimeout 146, 157
setmargins 146, 157
setmirrortype 146, 158, 159
setpagemargin 146, 158, 172
setpagemargin 146, 158, 172
setpageparams 146, 158, 172
setpagestackorder 146, 158
setpapertray 147
setprintername 146, 158
setresolution 146, 158
setsccebatch 146, 163, 164
setscinteractive 146, 147, 164
setsoftwareiomode 146, 159
settimeout 146, 159, 168
setuserdiskpercent 146, 147, 159
softwareiomode 146, 160
tumble 146, 160, 168
userdiskpercent 146, 147, 160
waittimeout 146, 160
configurationerror 18, 34, 51, 60, 61, 64, 65, 100, 106, 148, 166
Confirmation 20, 21, 27
ControlType 95
ControlPortNumber 76
Copies 21, 30, 31, 110, 113
CoverNote 21
CoverPagesSent 126
CoverSheet 20, 21, 22, 26, 27, 28, 29, 30
CoverSheetOnly 22, 31
CoverType 28
CurBufferType 41
CurDisplayList 41
CurFontCache 41
CurFormCache 41
CurInputDevice 42, 159, 160
CurOutlineCache 41
CurOutputDevice 42
CurPatternCache 41
currentdevparams operator 2, 37, 50, 97, 98, 110, 116, 124
currenthalftone operator 143
currentpagedevice operator 2, 5
CurrentPageNo 28
currentsystemparams operator 2, 37, 40
currentuserparams operator 2, 37
CurScreenStorage 41
CurSourceList 42
CurUPathCache 41
CutMedia 5

D
Darkness 108
DataBits 63, 162, 163
DataPortNumber 76
day 123
DefaultCaptionOn 25, 116
DefaultConfirmOn 21, 117
DefaultCoversOn 22, 31, 117
DefaultResolution 23, 117
DefaultRetryCount 24, 117
DefaultRetryInterval 26, 117
defaulttimeouts 146, 150
DeferredMediaSelection 6, 9
DelayedOutputClose 58, 62, 66, 68, 69, 71, 77, 79, 80, 81
deletejobsforall 124
details dictionaries 14
devdismount 147, 168
devforall 147, 169
devformat 147, 169
device parameters
Baud 62, 163
BlockSize 97, 101
BootDelay 106
BroadcastAddress 88
BSizeStandard 108
Bus 98
CartridgeID 102
CartridgeType 102
CheckParity 62, 163, 164
Checksum 87, 93
ConnectorType 95
ControlPortNumber 76
Darkness 108
DataBits 162, 163
DataPortNumber 76
DelayedOutputClose 58, 62, 66, 68, 69, 71, 77, 79, 80, 81
Enabled 59, 60, 61, 63, 65, 66, 68, 69, 71, 77, 79, 80, 81, 157
EthernetAddress 72, 93, 95
EtherTalkType 70, 72, 149
EtherTalkZone 72
Filtering 69, 72, 74, 77, 80, 82
FlowControl 63, 161, 163
Free 98, 150, 170
GatewayAddress 89
Handshake 67
HasNames 60, 64, 67, 68, 70, 72, 74, 77, 79, 81, 82, 97, 98, 102, 104, 170
InitializeAction 98, 99, 102, 104, 152, 169
InitiatorId 106, 107
Interleave 99
Interpreter 60, 61, 64, 65, 67, 69, 70, 73, 79, 81, 82, 110, 159
IPAddress 88, 89, 90, 93
IPAddressDynamic 88, 89, 90
LocalTalkType 70, 72, 149
LoginPassword 82
LogPriority 85
Mounted 98, 100, 101, 103, 104, 121, 169, 170
Name 95
NetworkAddress 93
NetworkMask 88, 90
NodeID 70
On 60, 61, 64, 67, 69, 71, 73, 75, 77, 80, 81, 82, 86, 88, 91, 92, 94, 95, 157
OutputDevice 67
PageCount 109
Parity 62, 64, 162, 163, 164
Physical 91, 94
PhysicalSize 100, 102, 103
Poll 106, 107
PrintHost 75, 78
PrivateHost 83
Protocol 60, 61, 63, 64, 65, 68, 159
ReceiveWindowSize 75, 78, 87, 88, 92
Removable 100, 103, 104, 170
Searchable 101, 103, 104, 170
SearchOrder 101, 103, 104, 170
SendWindowSize 75, 78, 87
StatusPortNumber 78
StopBits 66, 161, 163
StorageDevice 119
SysContact 83
SysLocation 84
SysName 84
TargetId 106, 107
TimeToStandby 109
TransmitEncapsulation 91, 94
TrapHost 84
Type 51, 66, 68, 69, 71, 73, 76, 79, 80, 81, 82, 85, 86, 87, 88, 91, 92, 94, 96, 101, 105, 107, 109, 122
Writeable 101, 105, 121, 150, 170
DeviceCMY 141
DeviceCMYK 141
DeviceGray 141
DeviceRenderingInfo 7
DeviceRGB 141
DeviceRGBK 141
devmount 147, 169
devstatus 147, 170
Diablo630 61
DiaLallee 18, 20, 22, 26, 125, 128
DiaLToneWaitPeriod 118, 122
diskonline 146, 150
diskstatus 146, 150
DoStartPage 43, 151, 156
doStartPage 146, 151
dosysstart 146, 151
Duplex 5, 110, 167, 168
duplexmode 146, 151, 167
Duration 128, 129

E

ECS 115
ECSDataWidth 115
11x17 147, 165
11x17tray 146, 160, 166
emulate 146, 147, 151
Emulator 139, 151
emulator parameters
AutoLF 115
BoldFontName 115
ColorSetup 114
Copies 110, 113
Duplex 110
ECS 115
ECSDataWidth 115
FontFixed 111, 113
FontHeight 111, 113
FontItalic 111, 113
FontNumber 111
FontPitch 111, 113
FontSource 111
FontSymbolSet 111, 114
FontTypeface 112, 114
FontWeight 112, 114
Landscape 114
LinesPerInch 114
LineWrap 112
ManualFeed 114
MaxLJMemory 112, 114
PaperSize 112
Pitch 115
RegFontName 115
TopMargin 113
Type 113, 114, 115
VMI 113
WaitTimeout 113, 114
Enabled 59, 60, 61, 63, 65, 67, 68, 69, 71, 74, 77, 79, 80, 81, 152, 157
Encoding 44, 138
EndPage 5
Envelopes
orientation in user space 33
EpsonFX850 61
ErrorArray 28, 125, 126, 128, 129
ErrorCorrect 23
ErrorIndex 28, 125, 126, 128, 129
EthernetAddress 72, 93, 95
EtherTalkType 70, 72
EtherTalkZone 72
ExitJamRecovery 7
exitserver 37, 49, 148, 150

F

FactoryDefaults 43
FatalErrorAddress 43
Fax 17, 142
fax device parameters
ActivityReport 116
DefaultCaptionOn 25, 116
<table>
<thead>
<tr>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>job dictionary entries</td>
</tr>
<tr>
<td>CallCount 125</td>
</tr>
<tr>
<td>Calls 125</td>
</tr>
<tr>
<td>DialCallee 125</td>
</tr>
<tr>
<td>ErrorArray 125</td>
</tr>
<tr>
<td>ErrorIndex 125</td>
</tr>
<tr>
<td>JobId 125</td>
</tr>
<tr>
<td>RecipientName 125</td>
</tr>
<tr>
<td>RecipientOrg 125</td>
</tr>
<tr>
<td>SenderName 126</td>
</tr>
<tr>
<td>SenderOrg 126</td>
</tr>
<tr>
<td>TimeBegan 126</td>
</tr>
<tr>
<td>TotalPages 126</td>
</tr>
<tr>
<td>TotalPagesSent 126</td>
</tr>
<tr>
<td>job records</td>
</tr>
<tr>
<td>deletejobsforall 124</td>
</tr>
<tr>
<td>jobsforall 124</td>
</tr>
<tr>
<td>reportjoblist 124, 125</td>
</tr>
<tr>
<td>returnjoblist 124, 125</td>
</tr>
<tr>
<td>JobId 125</td>
</tr>
<tr>
<td>JobName 39, 152</td>
</tr>
<tr>
<td>jobname 146, 152, 153</td>
</tr>
<tr>
<td>jobsforall 124</td>
</tr>
<tr>
<td>JobTimeout 40, 45, 150, 156, 157</td>
</tr>
<tr>
<td>jotimeout 146, 153</td>
</tr>
<tr>
<td>Jog 5</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>K</td>
</tr>
<tr>
<td>Kind 128, 129</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>Landscape 112, 114</td>
</tr>
<tr>
<td>LaserJetIII 60, 61</td>
</tr>
<tr>
<td>LaserJetIIIP 60, 61</td>
</tr>
<tr>
<td>ledger 147, 165</td>
</tr>
<tr>
<td>ledgerray 146, 153, 166</td>
</tr>
<tr>
<td>legal 147, 165</td>
</tr>
<tr>
<td>legaltray 146, 153, 166</td>
</tr>
<tr>
<td>letter 147, 165</td>
</tr>
<tr>
<td>lettersmall 147, 165</td>
</tr>
<tr>
<td>lettertray 146, 153, 166</td>
</tr>
<tr>
<td>LicenseID 45</td>
</tr>
<tr>
<td>limitcheck 164, 166</td>
</tr>
<tr>
<td>LimitPage 29</td>
</tr>
<tr>
<td>LinesPerInch 114</td>
</tr>
<tr>
<td>LineWrap 112</td>
</tr>
<tr>
<td>LocalLanguage 25, 119</td>
</tr>
<tr>
<td>LocalTalkType 70, 149</td>
</tr>
<tr>
<td>LogHost 85</td>
</tr>
<tr>
<td>LogicalSize 98, 99, 100, 102, 150, 152, 169, 170</td>
</tr>
<tr>
<td>LoginPassword 82</td>
</tr>
<tr>
<td>LogPriority 85</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>MailingTime 24, 129</td>
</tr>
<tr>
<td>ManualFeed 5, 114, 153</td>
</tr>
<tr>
<td>manualfeed 146, 153</td>
</tr>
<tr>
<td>ManualFeedTimeout 8, 150, 153</td>
</tr>
<tr>
<td>manualfeedtimeout 146, 153</td>
</tr>
<tr>
<td>ManualSize 141</td>
</tr>
<tr>
<td>instance of OutputDevice resource</td>
</tr>
<tr>
<td>category 141</td>
</tr>
<tr>
<td>Margins 9, 154, 157</td>
</tr>
<tr>
<td>margins 146, 154</td>
</tr>
<tr>
<td>MaxDictStack 39</td>
</tr>
<tr>
<td>MaxDisplayList 41</td>
</tr>
<tr>
<td>MaxExecStack 39</td>
</tr>
<tr>
<td>MaxFaxBuffer 119</td>
</tr>
<tr>
<td>MaxFontCache 41</td>
</tr>
<tr>
<td>MaxFontItem 39</td>
</tr>
<tr>
<td>MaxFormCache 41</td>
</tr>
<tr>
<td>MaxFormItem 39</td>
</tr>
<tr>
<td>MaxHWRenderingBuffer 45</td>
</tr>
<tr>
<td>MaxImageBuffer 46</td>
</tr>
<tr>
<td>MaxLJMemory 112, 114</td>
</tr>
<tr>
<td>MaxLocalVM 39</td>
</tr>
<tr>
<td>MaxOpStack 39</td>
</tr>
<tr>
<td>MaxOutlineCache 41</td>
</tr>
<tr>
<td>MaxPatternCache 41</td>
</tr>
<tr>
<td>MaxPatternItem 39</td>
</tr>
<tr>
<td>MaxRasterMemory 41, 46</td>
</tr>
<tr>
<td>MaxRetries 24, 117</td>
</tr>
<tr>
<td>MaxScreenItem 39</td>
</tr>
<tr>
<td>MaxScreenStorage 41</td>
</tr>
<tr>
<td>MaxSourceList 46</td>
</tr>
<tr>
<td>MaxUPathCache 41</td>
</tr>
<tr>
<td>MaxUPathItem 39</td>
</tr>
<tr>
<td>MediaColor 5</td>
</tr>
<tr>
<td>MediaPosition 9</td>
</tr>
<tr>
<td>MediaType 5</td>
</tr>
<tr>
<td>MediaWeight 5</td>
</tr>
<tr>
<td>MinBandBuffers 47</td>
</tr>
<tr>
<td>MinFontCompress 39</td>
</tr>
<tr>
<td>Minute 123</td>
</tr>
<tr>
<td>MirrorPrint 5</td>
</tr>
<tr>
<td>Month 123</td>
</tr>
<tr>
<td>Mounted 98, 100, 101, 103, 104, 121, 168, 169, 170</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>Name 95</td>
</tr>
<tr>
<td>NegativePrint 5</td>
</tr>
<tr>
<td>NetworkAddress 93</td>
</tr>
<tr>
<td>NetworkMask 88, 90</td>
</tr>
<tr>
<td>newsheet 146, 147, 154, 156, 157</td>
</tr>
<tr>
<td>node address 73</td>
</tr>
<tr>
<td>NodeID 70</td>
</tr>
<tr>
<td>note 147, 165</td>
</tr>
<tr>
<td>nPages 31</td>
</tr>
<tr>
<td>NumberOfCalls 29</td>
</tr>
<tr>
<td>NumCopies 5</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>O</td>
</tr>
<tr>
<td>On 60, 61, 64, 67, 69, 71, 73, 75, 77, 80, 81, 82, 86, 88, 91, 92, 93, 95, 157</td>
</tr>
<tr>
<td>Orientation 5</td>
</tr>
<tr>
<td>OutputAttributes 5, 18</td>
</tr>
<tr>
<td>OutputDevice 9, 17, 67, 138, 140, 165, 166</td>
</tr>
<tr>
<td>OutputDevice resource category</td>
</tr>
<tr>
<td>instances of</td>
</tr>
<tr>
<td>HWResolution 141</td>
</tr>
<tr>
<td>ManualSize 141</td>
</tr>
<tr>
<td>PageSize 141</td>
</tr>
<tr>
<td>ProcessColorModel 141</td>
</tr>
<tr>
<td>OutputFaceUp 5, 154, 158</td>
</tr>
<tr>
<td>OutputPage 9</td>
</tr>
<tr>
<td>OutputType 5</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>page device parameters</td>
</tr>
<tr>
<td>AdvanceDistance 5</td>
</tr>
<tr>
<td>AdvanceMedia 5</td>
</tr>
<tr>
<td>BeginPage 5</td>
</tr>
<tr>
<td>Bind 6</td>
</tr>
<tr>
<td>BindDetails 6</td>
</tr>
<tr>
<td>Booklet 6</td>
</tr>
<tr>
<td>BookletDetails 6</td>
</tr>
<tr>
<td>Collate 5</td>
</tr>
<tr>
<td>CutMedia 5</td>
</tr>
<tr>
<td>DeferredMediaSelection 6, 9</td>
</tr>
<tr>
<td>DeviceRenderingInfo 7</td>
</tr>
</tbody>
</table>
Duplex 5, 167, 168
EndPage 5
ExitJamRecovery 7
FaxOptions 7
Fold 7
FoldDetails 7
HWResolution 5, 9
ImageShift 7
ImagingBBox 5, 164, 165
InputAttributes 7, 18
InsertSheet 8
Install 5
Jog 5
ManualFeed 5, 153
ManualFeedTimeout 8, 150, 153, 156
Margins 9, 154, 157
MediaColor 5
MediaPosition 9
MediaType 5
MediaWeight 5
MirrorPrint 5
NegativePrint 5
NumCopies 5
Orientation 5
OutputAttributes 5, 18
OutputDevice 9, 17
OutputPage 5, 154, 158
OutputPage 9
OutputType 5
PageOffset 10
PageSize 5, 14, 33, 164, 165, 166
PageSize Policy 0 166
PageSize Policy 7 164
PageSize policy 7 14
PagesSent 29, 127
pagemstackorder 146, 154
PaperSize 112
Parity 62, 64, 162, 163, 164
Password 148
Pattern 44
PCL 60
Physical 91, 94
PhysicalSize 100, 102, 103
PixelBurst 142
Policies 5
Poll 106, 107
PostRenderingEnhance 10
PostRenderingEnhanceDetails 10
PostScript 61
PostScript Level 2 resources 2
PostScriptPassword 18, 19, 25, 30, 119
PreRenderingEnhance 10
PreRenderingEnhanceDetails 10
ProcessColorModel 10, 141
instance of OutputDevice resource
category 141
processcolors 146, 155
ProcInfo 25
ProcSet 44
product 146, 155
ProprinterXL 61
Protocol 60, 61, 63, 64, 65, 68, 159
ProtocolVersion 120

R

RamSize 48
ramsize 146, 155
rangecheck 35, 123, 148, 166
Raw 60, 61
RealFormat 41, 155
realformat 146, 155
receivejobsforall 128, 129
ReceivePostScript 18, 119, 120
ReceiveWindowSize 75, 78, 87, 92
Recipient 129
RecipientID 25
RecipientLanguage 25
RecipientMailStop 25
RecipientName 25, 26, 30, 125
RecipientOrg 26, 125
RecipientPhone 26, 30, 125
Regarding 26
related publications 3
Removable 100, 103, 104, 170
reportjoblist 124, 125
resources
Category 44, 140
ColorRendering 44, 138
ColorRenderingType 139
ColorSpace 44, 138
ColorSpaceFamily 139
Emulator 139, 151
Encoding 44, 138
FaxAdminOps 120, 124, 127
Filter 138
FMapType 139
Font 44, 137
FontType 139
Form 44, 138
FormType 139
Generic 44, 140
Halftone 44, 138
HalftoneType 139
HWOptions 138
ImageType 139
IODevice 50, 139
OutputDevice 138, 165, 166
Pattern 44, 138
PatternType 139
ProcSet 44, 138
RetriesLeft 129
RetriesTried 129
RetryInterval 26, 117
returnjoblist 124, 125
RevertToRaster 18, 19, 26, 30, 120
Revision 48, 156
revision 146, 156
Rings 120
Running 123

S

sccbatch 146, 163
sccinteractive 146, 156, 158, 163
Searchable 101, 103, 104, 170
SearchOrder 101, 103, 104, 170
Second 123
SenderID 26
SenderMailStop 26
SenderName 26, 27, 126
SenderOrg 27, 126
SenderPhone 27
SendPostScript 29
SendWindowSize 75, 78, 87
SeparationColorNames 11
SeparationOrder 11
Separations 5
ServiceEnable 121
setcolorscreen 39
setdefaulttimeouts 146
setdevparams operator 2, 37, 49, 50, 51, 60, 66, 97, 98, 101, 110, 116, 148
setdostartpage 146
setdosystart 146, 156
setduplexmode 146, 167
sethalftone operator 143
sethardwareiomode 146, 157
setjobtimeout 146, 157
setmargins 146
setpagedevice operator 2, 5, 7, 8, 9, 14, 17, 20, 26, 121, 145, 164, 165
setpagestackorder 146
setpapertray 147
setprintername 146, 158
setresolution 158
setscrbatch 146, 163, 164
setscrinteractive 146, 164
setsoftwareiomode 146, 159
setsystemparams operator 2, 37, 40, 43, 49, 148
settumble 146, 159, 168
setuserdiskpercent 146, 159
setuserparams operator 2, 37
showpage operator 21, 22
Signature 12
SlipSheet 12
softwareiomode 146, 160
Speaker 121
Staple 13
StapleDetails 13
startjob 37, 49, 148, 150
StartJobPassword 37, 38, 49, 148, 150
StartupMode 49, 151, 156
Status 128, 129
statusdict dictionary 145, 151, 165, 166
compatibility operators in 146
StatusPortNumber 78
StopBits 66, 161, 163
StorageDevice 119, 121
SysContact 83
SysLocation 84
SysName 84
system parameters
CreateTime 41, 149
ByteOrder 41, 149
CurBufferType 41
CurDisplayList 41
CurFontCache 41
CurFormCache 41
CurInputDevice 159, 160
CurOutlineCache 41
CurOutputDevice 42
CurPatternCache 41
CurScreenStorage 41
CurSourceList 42
CurStoredFontCache 42
CurStoredScreenCache 42
CurUPathCache 41
DoStartPage 43, 151, 156
FactoryDefaults 38, 43
FatalErrorAddress 43
FontResourceDir 44
GenericResourceDir 44, 45
GenericResourcePathSep 44
JobTimeout 45, 150, 156
LicenseID 45
MaxDiskFontCache 45
MaxDisplayList 41
MaxFontCache 41
MaxFormCache 41
MaxHWRenderingBuffer 45
MaxImageBuffer 46
MaxOutlineCache 41
MaxPatternCache 41
MaxRasterMemory 46
MaxScreenStorage 41
MaxSourceList 46
MaxStoredFontCache 46
MaxStoredScreenCache 47
MaxUPathCache 41
MinBandBuffers 47
PageCount 43, 47, 48, 154
Password 148
PrinterName 48, 70, 72, 155, 158
RamSize 48
RealFormat 41, 155
Revision 48, 156
StartJobPassword 37, 49, 148, 150
StartupMode 49, 151, 156
SystemParamsPassword 37, 38, 49, 120, 148, 150, 168, 169
ValidNV 49
WaitTimeout 50, 150, 156
systemdict dictionary 168
compatibility operators in 147
SystemParamsPassword 37, 38, 49, 120, 148, 150, 168, 169

T

TargetID 106, 107
TBCP 60, 61
Thresholds 144
Time 129
TimeBegan 126
timeout 40
TimeSent 29, 127
TimeToStandby 109
TODClock 142
TopMargin 113
TotalPages 126
TotalPagesSent 126
TransferFunction 144
TransmitEncapsulation 91, 94
transmitjobsforall 24, 127, 128
TrapHost 84
TraySwitch 13
Trim 13
TrimWhite 27, 30
tryphone operator 127
Tumble 5, 168
tumble 146, 160, 168
Type 51, 68, 69, 71, 73, 76, 79, 80, 81, 82, 85, 86, 87, 88, 91, 92, 94, 96, 101, 103, 105, 107, 109, 113, 114, 115, 122
entry in a Details dictionary 15
Type 6 halftone dictionary 143
entries
  HalftoneType 144
  Height 144
  Thresholds 144
  TransferFunction 144
  Width 144
Type1Coprocessor 142
typecheck 34
typecheck error 18

U
undefined 36
user parameters
  AccurateScreens 39
  JobName 39, 152
  JobTimeout 40, 153, 157
  MaxDictStack 39
  MaxExecStack 39
  MaxFontItem 39
  MaxFontItem 39
  MaxLocalVM 39
  MaxPatternItem 39
  MaxScreenItem 39
  MaxUPathItem 39
  MinFontCompress 39
  VMReclaim 39
  VMThreshold 39
  WaitTimeout 40, 160
userdict dictionary 164
  compatibility operators in 147
userdiskpercent 146, 147, 160
USModem 142

V
ValidNV 49
VMI 113
VMReclaim 39
VMThreshold 39

W
WaitForDialTone 122
WaitTimeout 40, 50, 113, 114, 150, 156, 160
waittimeout 146, 160
Width 144
WorldModem 142