Technical support personnel at Linotype-Hell are asked many questions related to the PostScript™ page description language, but undoubtedly the most common request that they get is for lists of PostScript errors. To that end, reprinted in this article is a list of PostScript errors from Adobe Systems 'Red' book, *The PostScript Language Reference Manual* (Addison Wesley). But there is much more to troubleshooting than lists of errors and solutions. If lists were the only thing required for troubleshooting, then PostScript errors would be easily solved in every case. In fact, cause/solution lists can often be deceptive rather than helpful. Why? Because a given error may have many possible causes, and any list that is created is locked in time based on the PostScript version, the software application version, hardware revisions, and many other variables. As technology changes the list is soon outmoded.

It is more important to teach people how to approach PostScript error troubleshooting. That was the purpose of the article, *Troubleshooting PostScript Errors*, which is included in the 1992 Linotype-Hell technical information notebook. Still, there is no way of getting around the fact that the most advanced troubleshooting methods require that the troubleshooter have some familiarity with PostScript programming. For that reason, a number of PostScript programming resources are listed at the end of this document.

Error types

There are three possible types of errors in a PostScript environment[1]:

Application and printer driver errors are commonly mentioned to Linotype-Hell technical support staff. Unfortunately, these errors are based on lists of errors that have been compiled in the driver and they tell you very little about the actual problem. Common application and printer driver errors are: -4100, -8133, or 'The job is OK but can't be printed.' These errors do not help very much in troubleshooting.

[1]This topic is covered in greater detail in *Troubleshooting PostScript Errors* which appears in the 1992 Linotype-Hell technical information notebook.

RIP reported errors are the errors that appear on the panel of a Raster Image Processor (RIP) or imagesetter. Probably the most common Linotype-Hell RIP error is E12 (no take-up cassette). These errors are listed in the RIP user manual and are usually very easily solved.

PostScript interpreter errors are reported back from the RIP after a job has been processed. Finding these errors is the first step in troubleshooting.

Application and printer driver errors may be intertwined with PostScript interpreter errors in the following fashion. Sometimes the actual PostScript interpreter error is intercepted by the driver and replaced with an application or printer driver error. The PostScript interpreter error may appear on the screen only very briefly or may be thrown away entirely.

PostScript error list

There are thirty possible PostScript errors. (See chart on the following page.) Some of the errors are so obscure that you are unlikely ever to see them. Most have been PostScript errors since the inception of the language. Some have been added with the introduction of Level 2 PostScript (configurationerror, invalidcontext, invalidid, undefinedresource) and one (dictfull) will no longer be an error in PostScript Level 2.

The 30 kinds of PostScript errors:

PostScript error	*Red Book explanation*	*Additional explanation/typical occurrence*
configurationerror	setpagedevice request cannot be satisfied	i.e., a requested feature that is not available on that device
dictfull	no more room in dictionary	(see description on page 7)
dictstackoverflow	too many begins	each begin command must have a corresponding end command
dictstactunderflow	too many ends	each end command must have a corresponding begin command
execstackoverflow	exec nesting too deep	too many pending nested procedure calls (very obscure error)
handleerror	called to report error information	(not actually an error procedure)
interrupt	external interrupt request (e.g., Control-C)	Control-C is used to abort a program
invalidaccess	attempt to violate access attribute	i.e., writing to a read-only dictionary
invalidcontext	improper use of context operation	(Display PostScript error)
invalidexit	exit not in loop	exit command improperly used (very obscure error)
invalidfileaccess	unacceptable access string	i.e., writing to a read-only file (very obscure error)
invalidfont	invalid font name or dictionary	corrupted or improperly formed font dictionary
invalidid	invalid identifier for external object	(Display PostScript error)
invalidrestore	improper restore	often a string, dictionary, or procedure is left on the stack that needs to be discarded before restore
ioerror	input/output error occurred	may signal incorrect handshaking protocol or improperly terminated or faulty cables
limitcheck	implementation limit exceeded	i.e., with too many path elements in a lineto or curveto segment
nocurrentpoint	current point is undefined	i.e., if the required moveto command is not supplied
rangecheck	operand out of bounds	i.e., a film width request greater than that used by the target output device
stackoverflow	operand stack overflow	received more data than it can accommodate
stackunderflow	operand stack underflow	expected more data than it received
syntaxerror	PostScript language syntax error	i.e., an open or closing mark (for example, a bracket) is missing
timeout	time limit exceeded	occurs when there is too great a time gap between receiving two portions of the same job
typecheck	operand of wrong type	i.e., an operator expected one type of data and got something else (very frequent error)
undefined	name not known	no definition exists in the dictionary
undefinedfilename	file not found	like it says
undefinedresource	resource instance not found	Level 2 error (very obscure error)
undefinedresult	over/underflow or meaningless result	often a division by zero
unmatchedmark	expected mark not on stack	such as a closed bracket to go with an open one
unregistered	internal error	undefined, undocumented statusdict operator (very obscure error)
VMerror	VM exhausted	no more available user memory

PostScript error format

The format of PostScript errors from the RIP always consists of two parts: the type of error, and the offending command. Any other message formats are generated either by the application or printer driver. The message coming back from the RIP will usually look like this:

```
%%[ Error:           OffendingCommand:     ]%%
```

Here's an example:

```
%%[ Error: dictfull; OffendingCommand: def ]%%
```

In the above example, the type of error is *dictfull* and *def* is the offending command. Generally, a dictfull error occurs when the PostScript program attempts to define a new entry in a dictionary that is already full. A dictionary has a fixed limit on the number of entries that it can hold, and this limit is established when the dictionary is created. (Dictionaries are created to hold pairs of PostScript objects, for example a variable and its current value.)

The offending command is the command that was being executed when the error occurred. In this example, the def operator tells the interpreter to define a name (called a key) and associate it with a value (which can be a number, a procedure, or even a dictionary). The def command almost always appears with a dictfull error because the interpreter is usually in the process of defining something when it discovers that there is no more room in the dictionary. The dictfull error is very common, and often occurs because of problems related to the software application:

- The software application may create too small a dictionary and then proceed to create to many entries for that dictionary. Or, other applications may use the dictionary in ways that the first software application programmers didn't anticipate.[2]
- The software application may use an existing dictionary, in most cases the userdict. The userdict is a device dependent dictionary containing spaces for 200 entries. The number of entries actually used will vary with different output devices (and PS versions). This is one common reason why files that run on a laser printer may not run to a high resolution output device: the laser printer's userdict may be nearly empty, and is not overtaxed by applications that use it. The situation is very different in the userdicts of more sophisticated output devices. Certain keys have to be placed in the userdict by the raster image processor (RIP) running the imagesetter. This leaves fewer available spots in these userdicts for a software application that might choose to use it.
- Because there are a number of different dictionaries that may be placed on the dictionary stack[3], there may be some problem when a PostScript program adds to its dictionary. For example, errors often occur because the target dictionary is not on the top of the dictionary stack. If the same key occurs in two dictionaries, the RIP will take the first occurrence of the key that it finds (and that may not be the one that was redefined).

The dictfull error is a good example of how changes in the PostScript page description language change the nature of the errors that users see. Engineers at Adobe Systems, concerned by problems with dictionaries with fixed limits, decided to integrate a method of increasing dictionary size on-the-fly into PostScript Level 2. By increasing the size of a dictionary as it starts to fill up, dictfull errors can be avoided. This is one of many changes that have been made in PostScript Level 2 and is the reason why dictfull errors will not appear in PostScript Level 2 devices.

The first two columns from the chart on page 6 are from page 359 of the PostScript Language Reference Manual, Second Edition, Adobe Systems, Inc., Addison-Wesley Publishing Co., Inc., 1990. More information on the meaning of each error listed here is shown in the Operator Details section from page 360-553. The 'Additional explanation/typical occurrence' column is drawn from a variety of sources including the PostScript Language Level 2 Reference Card (see list of recommended materials).

[2]This is one more reason why lists of errors can be misleading. For example, new dictful errors may occur with files that have previously run without error if software applications are used together in a new fashion, or even if there are releases of new versions of software. So-called well-formed or well-behaved applications (those that follow the PostScript Document Structuring Conventions) should not cause this kind of problem, but in the real world, not all applications are well-formed.

[3]A stack is a holding place for all different types of data. There are several stacks that are used in PostScript, including one for dictionaries.

Troubleshooting

Troubleshooting is the process of finding out what the error is, and then once the error is known, doing something about it. There are a number of different levels of complexity in troubleshooting, some requiring diagnostic tools or programming experience, and some being quite simple.

Divide and Conquer is a simple strategy that narrows down the location of a PostScript error by removing or simplifying page elements until the job prints error free. (For a fuller description see *Troubleshooting PostScript Errors*.) While Divide and Conquer is a useful technique, it is also very time consuming. However, it requires no familiarity with PostScript programming.

Error handling programs are primarily helpful in finding the location of an error, not in the solution. But identifying the error is a very important first step towards solving the problem. In addition, error handling programs have other useful features. For example they may print out the error on the film, or save it in a location that makes it easier to find. They may show you the next 2-4 lines of code, or even the current point and other current page parameters.

Finally, the most sophisticated solution to PostScript errors involves going into a file and making changes to the code. Using **PostScript programming** in this manner is an important tool, but one that requires an experienced eye and obviously, some amount of programming skill. (LaserTalk™ is very useful for troubleshooting.[4] It allows you to download a file line by line and view any problems as they develop. It also lets you keep track of the stacks and other parameters or current conditions during execution of a file.)

[4]Adobe Systems, Inc., the maker of LaserTalk, is in the process of producing an updated version.

Analyzation programs

While not strictly speaking an error handling tool, analyzation programs like LinoCheck™ and LaserCheck™ allow users to identify common problems before they reach film. LinoCheck is a Linotype-Hell product and LaserCheck is a product developed by Systems of Merritt (see below).

Conclusion

These are the two most important troubleshooting tips:

- Learn to identify the PostScript error. (Having the real error message rather than an application or printer driver message makes it easier to proceed with troubleshooting.)
- Learn more about the PostScript page description language.

Recommended materials

Here are some good sources of information on PostScript programming:

- Two books by Glenn Reid provide excellent and very readable information on PostScript. They are *Thinking in PostScript* and *PostScript Language Program Design* (also known as the 'Green book'). Both books are published by Addison-Wesley.
- Frank Braswell of Systems of Merritt has produced a handy, plastic-coated, one page reference sheet of PostScript operators, interpreter messages, and errors. This is a nice tool for the serious troubleshooter. Braswell also offers a two-day seminar on PostScript error troubleshooting, call him at 205-660-1240 for more information.
- Acquired Knowledge offers a variety of classes on PostScript programming, call them at 619-587-4668 for more information.

Acknowledgements

Many thanks to Frank Braswell of Systems of Merritt, Bob Schaffel executive director of the Professional Prepress Alliance, and Gus Barbuto of Linotype-Hell Company for their help in producing this document.

Please direct any questions or comments to: Jim Hamilton, Marketing Department, Linotype-Hell Company, 425 Oser Avenue, Hauppauge, NY 11788
(For subscription information on the Linotype-Hell technical information series, please call 1-800-842-9721.)

February 1993, Part Number 7006