CHAPTER

# 10

# Optimizing System Performance



## Overview

All software users are concerned about system speed and resource consumption. This section provides suggestions that may increase performance and reduce resource consumption in the OpenVMS operating environment. Suggestions are grouped by SAS System function: data set I/O, external file I/O, and system start-up.

SAS System performance partially depends on the performance of the underlying OpenVMS environment. This document does not discuss how to improve the performance of your OpenVMS environment, but it does offer some suggestions, such as installing SAS images, that can aid OpenVMS system-wide performance.

*Note:* This section does not discuss OpenVMS hardware solutions to problems that are related to I/O. △

The suggestions described here are based on testing on an OpenVMS Alpha workstation with 64MB of memory and an attached SCSI disk drive. Both SAS performance data that is generated by the STIMER system option and information from the DCL SHOW STATUS command were used to collect data points.

Each suggestion includes the following information in a tabular summary at the beginning of each section:

| | |
|---|---|
| Job type | is the type of SAS application affected. |
| User | is the person(s) who can implement the suggestion. |
| Usage | is a usage example. |
| Benefit | describes the potential benefits. |
| Cost | is the cost in system performance. |

Use this information to help you decide which suggestions are suitable for your SAS System applications.

Most of these suggestions have an associated cost, which can often involve a tradeoff between resources. For example, reduced I/O may require more memory consumption or greater CPU time. Pay careful attention to the possible costs of each suggestion. Some suggestions can cause performance degradation if they are misapplied.

# Data Set I/O

The information that is presented in this section applies to reading and writing SAS data sets. In general, the larger your data sets, the greater the potential performance gain for your entire SAS job. The performance gains that are described here were observed on data sets of approximately 100,000 blocks.

## Allocating Data Set Space Appropriately

| | |
|---|---|
| Job type | Jobs that write data sets. |
| User | SAS programmer. |
| Usage | Use ALQ=*x* and DEQ=*y* (or ALQMULT=*x* and DEQMULT=*y*) as LIBNAME statement options or as data set options, where *x* and *y* are values representing the number of blocks. |
| Benefit | There is up to a 50-percent decrease in elapsed time on write operations as reflected in fewer direct I/Os. File fragmentation is also reduced, thereby enhancing performance when you read the data set. |
| Cost | You will experience performance degradation when ALQ= or DEQ= values are incompatible with data set size |

In Version 8, the SAS System initially allocates enough space for 10 pages of data for a data set. Each time the data set is extended, another 5 pages of space is allocated on the disk. OpenVMS maintains a bit map on each disk that identifies the blocks that are available for use. When a data set is written and then extended, OpenVMS alternates between scanning the bit map to locate free blocks and actually writing the data set.

However, if the data sets were written with larger initial and extent allocations, then write operations to the data set would proceed uninterrupted for longer periods of time. At the hardware level, this means that disk activity is concentrated on the data set, and disk head seek operations that alternate between the bit map and the data set are minimized. The user sees fewer I/Os and faster elapsed time.

Large initial and extent values can also reduce disk fragmentation. SAS data sets are written using the RMS algorithm "contiguous best try." With large preallocation, the space is reserved to the data set and does not become fragmented as it does when inappropriate ALQ= and DEQ= values are used.

SAS Institute recommends setting ALQ= to the size of the data set to be written. If you are uncertain of the size, underestimate and use DEQ= for extents. Values of DEQ= larger than 5000 blocks are not recommended. For information about predicting data set size, see "Estimating the Size of a SAS Data Set" on page 131.

The following is an example of using the ALQ= and DEQ= options:

```
libname x '[]';
/* Know this is a big data set. */
data x.big (alq=100000 deq=5000);
   length a b c d e f g h i j k l m
_n o p q r s t u v w x y z $200;
   do ii=1 to 13000;
   output;
end;
run;
```

*Note:* If you do not want to specify an exact number of blocks for the data set, use the ALQMULT= and DEQMULT= options. △

### References for Allocating Data Set Space

- Data set option: "ALQ=" on page 249
- Data set option: "DEQ=" on page 256
- Data set option: "ALQMULT=" on page 250
- Data set option: "DEQMULT=" on page 257
- *Guide to OpenVMS File Applications*

## Turning Off Disk Volume Highwater Marking

| | |
|---|---|
| Job type | Any SAS application that writes data sets. Data set size is not important. |
| User | System manager. |
| Usage | Use the /NOHIGHWATER_MARKING qualifier when initializing disks. For active disks, issue the DCL command SET VOLUME/NOHIGHWATER_MARKING. |
| Benefit | There is a greater percentage gain for jobs that are write intensive. The savings in elapsed time can be as great as 40 percent. Direct I/Os are reduced. |

| | |
|---|---|
| Cost | There is no performance penalty. However, for security purposes, some OpenVMS sites may require this OpenVMS highwater marking feature to be set. |

Highwater marking is an OpenVMS security feature that is enabled by default. It forces prezeroing of disk blocks for files that are opened for random access. All SAS data sets are random-access files and, therefore, pay the performance penalty of prezeroing, increased I/Os, and increased elapsed time.

Two DCL commands can be used independently to disable highwater marking on a disk. When initializing a new volume, use the NOHIGHWATER_MARKING qualifier to disable the highwater function as in the following example:

```
$ initialize/nohighwater $DKA470 mydisk
```

To disable volume highwater marking on an active disk, use a command similar to the following:

```
$ set volume/nohighwater $DKA200
```

### References for Turning Off Disk Volume Highwater Marking

- *OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems*
- *OpenVMS DCL Dictionary A-M*

## Eliminating Disk Fragmentation

| | |
|---|---|
| Job type | Any jobs that frequently access common data sets. |
| User | SAS programmer and system manager. |
| Usage | Devote a disk to frequently accessed data sets, or keep your disks defragmented. |
| Benefit | The savings in elapsed time varies with the current state of the disk, but it can exceed 50 percent on write operations and 25 percent on read operations. |
| Cost | The cost to the user is the time and effort to better manage disk access. For the system manager, it can involve regularly defragmenting disks or obtaining additional disk drives. |

Any software that reads and writes from disk benefits from a well-managed disk. This applies to SAS data sets. On an unfragmented disk, files are kept contiguous; thus, after one I/O operation, the disk head is well positioned for the next I/O operation.

A disk drive that is frequently defragmented can provide performance benefits. Use a frequently defragmented disk to store commonly accessed SAS data sets. In some situations, adding an inexpensive SCSI drive to the configuration allows the system manager to maintain a clean, unfragmented environment more easily than using a large disk farm. Data sets maintained on an unfragmented SCSI disk may perform better than heavily fragmented data sets on larger disks.

By *defragmenting*, we mean a process that runs the OpenVMS Backup Facility after regular business hours. SAS Institute does not recommend using dynamic defragmenting tools that run in the background of an active system because such programs can corrupt files.

# Setting Larger Buffer Size for Sequential Write and Read Operations

| | |
|---|---|
| Job type | SAS steps that do sequential I/O operations on large data sets. |
| User | SAS programmer. |
| Usage | The CACHESIZ= data set option controls the buffering of data set pages during I/O operations. CACHESIZ= can be used either as a data set option or in a LIBNAME statement that uses the BASE engine. The BUFSIZE= data set option sets the data set page size when the data set is created. BUFSIZE= can be used as a data set option, in a LIBNAME statement, or as a SAS system option. |
| Benefit | There is as much as a 30-percent decrease in elapsed time in some steps when an appropriate value is chosen for a particular data set. |
| Cost | If the data set observation size is large, substantial space in the data set may be wasted if you do not choose an appropriate value for BUFSIZE=. Also, memory is consumed for the data cache, and multiple caches may be used for each data set opened. |

## Using the BUFSIZE= Option

The BUFSIZE= data set option sets the SAS internal page size for the data set. Once set, this becomes a permanent attribute of the file that cannot be changed. This option is meaningful only when you are creating a data set. If you do not specify a BUFSIZE= option, the SAS System selects a value that contains as many observations as possible with the least amount of wasted space.

An observation cannot span page boundaries. Therefore, unused space at the end of a page may occur unless the observations pack evenly into the page. By default, the SAS System tries to choose a page size between 8192 and 32768 if an explicit BUFSIZE= option has not been specified. If you increase the BUFSIZE= value, more observations can be stored on a page, and the same amount of data can be accessed with fewer I/Os. When explicitly choosing a BUFSIZE, be sure to choose a value that does not waste space in a data set page, resulting in wasted disk space. The highest recommended value for BUFSIZE= is 65024.

The following is an example of an efficiently written large data set, using the BUFSIZE= data set option. Note that in the following example, BUFSIZE=63488 becomes a permanent attribute of the data set:

```
libname buf '[]';
data buf.big (bufsize=63488);
   length a b c d e f g h i j k l m
          n o p q r s t u v w x y z $200;
   do ii=1 to 13000;
   output;
end;
run;
```

## Using the CACHENUM= Option

For each SAS file that you open, the SAS System maintains a set of caches to buffer the data set pages. The size of each of these caches is controlled by the CACHESIZ= option. The number of caches used for each open file is controlled by the CACHENUM= option. The ability to maintain more data pages in memory potentially reduces the

number of I/Os that are required to access the data. The number of caches that are used to access a file is a temporary attribute. It may be changed each time you access the file.

By default, up to 10 caches are used for each SAS file that is opened; each of the caches is the value (in bytes) of CACHESIZ= in size. On a memory-constrained system you may wish to reduce the number of caches used in order to conserve memory.

The following example shows using the CACHENUM= option to specify that 8 caches of 65024 bytes each are used to buffer data pages in memory.

```
proc sort data=cache.big (cachesiz=65024 cachenum=8);
   by serial;
run;
```

### Using the CACHESIZ= Option

The SAS System maintains a cache that is used to buffer multiple data set pages in memory. This reduces I/O operation by enabling SAS to read or write multiple pages in a single operation. SAS maintains multiple caches for each data set that is opened. The CACHESIZ= data set option specifies the size of each cache.

The CACHESIZ= value is a temporary attribute that applies only to the data set that is currently open. You can use a different CACHESIZ= value at different times when accessing the same file. To conserve memory, a maximum of 65024 bytes is allocated for the cache by default. The default allows as many pages as can be completely contained in the 65024-byte cache to be buffered and accessed with a single I/O. However, you can specify a CACHESIZ= value of up to 65024 bytes, the largest amount that can be accessed in a single I/O in an OpenVMS operating environment.

Here is an example that uses the CACHESIZ= data set option to write a large data set efficiently. Note that in the following example, CACHESIZ= value is *not* a permanent attribute of the data set:

```
libname cache '[]';
data cache.big (cachesiz=65024);
   length a b c d e f g h i j k l m
          n o p q r s t u v w x y z $200;
   do ii=1 to 13000;
   output;
end;
run;
```

## Using Asynchronous I/O When Processing SAS Data Sets

| | |
|---|---|
| Job type | Jobs that read or write SAS files. |
| User | SAS programmer. |
| Usage | The BASE engine now performs asynchronous reading and writing by default. This allows overlap between SAS data set I/O and computation time.<br><br>*Note:* Asynchronous reading and writing is enabled only if caching is turned on. △ |
| Benefit | Asynchronous I/O allows other processing to continue while waiting on I/O completion. If there is a large gap between the CPU time used and the elapsed time reported in the FULLSTIMER statistics, asynchronous I/O can help reduce that gap. |

| | |
|---|---|
| Cost | Because data page caching must be in effect, the memory usage of the I/O cache must be incurred. For more information about controlling the size and number of caches used for a particular SAS file, see the data set options "CACHENUM=" on page 253 and "CACHESIZ=" on page 253. |

In Version 8, asynchronous I/O is enabled by default. There are no additional options that need to be specified to use this feature. For all SAS files that use a data cache, SAS performs asynchronous I/O. Since multiple caches are now available for each SAS file, while an I/O is being performed on one cache of data, the SAS System may continue processing using other caches. For example, when SAS writes to a file, once the first cache becomes full an asynchronous I/O is initiated on that cache, but the SAS System does not have to wait on the I/O to complete. While that transaction is in progress, the SAS System can continue processing new data pages and store them in one of the other available caches. When that cache is full, an asynchronous I/O may be initiated on that cache as well.

Similarly, when SAS reads a file, additional caches of data may be read from the file asynchronously in anticipation of those pages being requested by the SAS System. When those pages are required, they will have already been read from disk, and no I/O wait need occur.

Because caching (with multiple caches) needs to be enabled in order for asynchronous I/O to be effective, if the cache is disabled with the CACHESIZ=0 option or the CACHENUM=0 option, no asynchronous I/O can occur.

### References for Using Asynchronous I/O

- Data set option: "CACHENUM=" on page 253
- Data set option: "CACHESIZ=" on page 253

## External I/O

The following guidelines apply to reading and writing OpenVMS native files using the SAS System. For several of the suggestions, the larger your files, the more performance gain for your entire SAS job. These suggestions parallel several of the SAS data set I/O suggestions.

### Allocating File Space Appropriately

| | |
|---|---|
| Job type | SAS procedures and DATA steps that write external files. |
| User | SAS programmer. |
| Usage | The ALQ= and DEQ= options are specified as part of the FILENAME or FILE statement. |
| Benefit | Specifying appropriate values can decrease elapsed time up to 50 percent and reduce disk fragmentation. |
| Cost | You will experience performance degradation when ALQ= and DEQ= values are incompatible with file size. |

The SAS System allocates disk space for external files based on the value of ALQ=. It then extends the file if needed, based on the value of DEQ=. By default, the ALQ=

value is 0 (indicating that the minimum number of blocks required for the given file format is used) and the value for DEQ= is 0 (telling OpenVMS RMS to use the process's default value). For more information about specifying the ALQ= and DEQ= options in a FILENAME or FILE statement, see "FILENAME" on page 357 and "FILE" on page 355.

Every time a file must be extended, the system must search the disk for free space. This requires I/Os. When this is done repeatedly for large files, performance degrades. By setting larger ALQ= and DEQ= values for large files, this overhead will be reduced. Optimal I/O will occur when ALQ= is equal to the size of the file. Because this is not always feasible, it is better to underestimate the value for ALQ= and set a larger DEQ= value. This allocates enough space for a smaller file, while extending it occasionally to meet the demands of a larger file. Allocating too much space may be costly if /HIGHWATER_MARKING is set on the disk. (For more information, see "Turning Off Disk Volume Highwater Marking" on page 210.)

### References for Allocating File Space

- *Guide to OpenVMS File Applications*
- Statement: "FILE" on page 355
- Statement: "FILENAME" on page 357

## Turning Off Disk Volume Highwater Marking

| | |
|---|---|
| Job type | Jobs that write external files. |
| User | System manager. |
| Usage | Use the /NOHIGHWATER_MARKING qualifier when initializing disks. For active disks, issue the DCL command SET VOLUME/NOHIGHWATER_MARKING. |
| Benefit | Elapsed time can be improved by up to 40 percent. Direct I/Os are reduced. |
| Cost | There is no performance penalty. However, for security purposes, some OpenVMS sites may require this OpenVMS highwater marking feature to be set. |

The SAS System uses the random access method when opening external files. This means that allocated disk space does not have to be processed in a sequential method. /HIGHWATER_MARK is a safeguard that clears disk space before it is allocated to remove residue of former files. To do this, the entire space allocated has to be overwritten. Overwriting the space costs some elapsed time and I/Os. If the data that are stored on the disk are not of a truly confidential nature, then a performance gain can be achieved by disabling highwater marking on this disk.

Two DCL commands can be used independently to disable highwater marking on a disk. When initializing a new volume, use the following to disable the highwater function:

```
$ initialize/nohighwater $DKA470 mydisk
```

To disable volume highwater marking on an active disk, use a command similar to the following:

```
$ set volume/nohighwater $DKA200
```

### References for Turning Off Disk Volume Highwater Marking

□ *OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems*

## Eliminating Disk Fragmentation

| | |
|---|---|
| Job type | Any jobs that access common external files frequently. |
| User | System manager. |
| Usage | You will need to devote a disk to frequently accessed files or keep your disks defragmented. |
| Benefit | The savings on elapsed time depend on the current state of the disk, but the time can be reduced by up to 40 percent. |
| Cost | The cost to the user is the time and effort to better manage disk access rather than letting the OpenVMS environment do all of the work. For the system manager, this may involve regularly defragmenting disks or obtaining additional disk drives. |

On an unfragmented disk, files are contiguous, so after one I/O operation the disk head is well positioned for the next I/O operation. Split I/Os are rare on an unfragmented disk, which decreases elapsed time to perform I/O.

Where possible, dedicating a disk drive to frequent defragmentation can provide performance benefits. Use this disk to store commonly accessed SAS external files. In some situations, adding an inexpensive SCSI drive to the configuration may allow the system manager to maintain a clean, unfragmented environment more easily than maintaining a large disk farm. Files that are maintained on this unfragmented SCSI disk may perform better than heavily fragmented files on larger disks.

## Specifying Default Multiblock Count

| | |
|---|---|
| Job type | Jobs that write large external files. |
| User | SAS programmer. |
| Usage | The MBC= option (multiblock count) is specified as part of the FILENAME or FILE statement. |
| Benefit | Elapsed time may be improved by 25 to 35 percent on jobs that output large external files. |
| Cost | Increasing multiblock count may slightly increase requirements for memory. |

By default, the SAS System uses the default value for your process for the multiblock count specified by the RAB$B MBC field in OpenVMS RMS. You can use the MBC= external I/O option to specify the size of the I/O buffers that OpenVMS RMS allocates for writing or reading an external file. The MBC= option controls how many 512-byte pages of memory are reserved to perform file access. When you increase the buffer size, you use more memory.

We recommend a value of approximately 32 blocks for the MBC= option when you are writing a very large external file (over 100,000 blocks). You may see improvement

in elapsed time up to 35 percent. Only minimal gains in performance will occur when you specify the MBC= option for reading an external file.

*Note:* You can use the MBC= option to affect a particular file. If you want to specify a default multiblock count for your process that will affect all external files in your SAS program, use the DCL SET RMS_DEFAULT command. △

# System Start-up

| | |
|---|---|
| Job type | All jobs. |
| User | System manager. |
| Usage | The OpenVMS Install facility can be used to make core SAS images resident in memory. |
| Benefit | Elapsed time of system start-up may decrease as much as 30 percent when running the SAS System under X-windows and by 40 percent in other modes. |
| Cost | With Version 8 of the SAS System, installing all images listed in the sample commands consumes between 7500 and 1200 global pages and 6 to 8 global sections. Refer to the list below for the difference in cost for VAX and AlphaVMS. |

Installing SAS System images can decrease the elapsed time of SAS System start-up by up to 40 percent. Installing images is most effective on systems where two or more users are using SAS simultaneously. Use the following commands in the system start-up file to install the core set of SAS images:

```
$ @SASdisk:[SAS8.TOOLS]SAS8_SYSTEM.COM
$ INSTALL :== $SYS$SYSTEM:INSTALL/COMMAND"
$ INSTALL ADD SAS$ROOT:[IMAGE]SAS8.EXE/OPEN/SHARE
$ INSTALL ADD SAS$ROOT:[PROCS]SASBXSPH.EXE/OPEN/SHARED
$ INSTALL ADD SAS$ROOT:[PROCS]SASSDS.EXE/OPEN/SHARED
$ INSTALL ADD SAS$ROOT:[PROCS]SASMOTIF.EXE/OPEN/SHARED
$ INSTALL ADD SAS$ROOT:[PROCS]SASMSG.EXE/OPEN/SHARED
```

*SASdisk* is the disk containing the SAS System; *SAS$ROOT* is the directory of the SAS System.

The Install facility can be used to make core SAS images resident in the memory. Elapsed time of system start-up may decrease by as much as 30 percent when running the SAS System under X-windows, and by 40 percent in other modes. With Version 8 of the SAS System, installing the images listed in the sample commands consumes the following resources:

VAX

| | |
|---|---|
| SAS8.EXE | 3 global sections and 1246 global pages |
| SASBXSPH.EXE | 1 global section and 3028 global pages |
| SASDS.EXE | 1 global section and 74 global pages |
| SASMOTIF.EXE | 1 global section and 3274 global pages |

AlphaVMS

| | |
|---|---|
| SAS8.EXE | 2 global sections and 1680 global pages |
| SASBXSPH.EXE | 2 global sections and 5168 global pages |
| SASDS.EXE | 2 global sections and 112 global pages |
| SASMOTIF.EXE | 2 global sections and 4320 global pages |

### References for System Start-up

- *OpenVMS System Manager's Manual: Tuning, Monitoring, and Complex Systems*
- *Installation Instructions and System Manager's Guide for the SAS System under OpenVMS Alpha and VAX, Version 8*

## Optimizing Memory Usage

You can make tradeoffs between memory and other resources. This is explained in "Data Set I/O" on page 204 and "External I/O" on page 209. To make the most of the I/O subsystem, you need to use more, larger buffers. These buffers are allocated out of your available virtual address space and must share your physical address space (that is, your working set) with the other memory demands of your SAS session.

Therefore, optimization of other resources is often at the expense of using more and more memory. If memory is your critical resource, there are several things you can do to reduce the dependence on increased memory. However, most of these techniques are at the expense of increased I/O processing or increased CPU usage.

Increasing the values of the MBC= and MBF= external I/O statement options, the CACHESIZ= option, and the BKS= option enables you to optimize the I/O subsystem by reducing the number of accesses necessary to reference the data. But reducing the number of accesses uses more memory. If you are operating in a memory-constrained environment, you need to reduce these values to minimize memory usage.

The size of your I/O buffers depends on your working set size. It is important not to increase I/O buffers up to the point that you exhaust your available working set. If you do this, you will notice an increase in the page fault rate for your job.

### Using the LOADLIST= System Option

Another way to optimize memory usage is to ensure that images that are used most often are installed as known images to the OpenVMS operating environment. If you use the LOADLIST= system option, the SAS System reports which images were used most often. If you install the most heavily used images as known images, they do not require additional physical memory if used by multiple users. For information about using the INSTALL utility, refer to *Installation Instructions and System Manager's Guide for the SAS System, Release 8.01, under OpenVMS*. For information about the LOADLIST= system option, see "LOADLIST=" on page 418.

### Using the UNLOAD System Option

A final technique to improve memory usage is to ensure that you make proper use of the UNLOAD system option. If your application performs many unique tasks, turn the

UNLOAD system option on. Then, after each PROC and DATA step ends and after each window is brought down in the windowing environment, the SAS System purges from memory the SAS images associated with the step that just ended. Thus, memory is freed for use in the next step. In applications that perform the same tasks over and over, specify NOUNLOAD.

Specifying NOUNLOAD keeps images in memory. This strategy is effective for jobs that run the same small set of procedures again and again, and SAS System performance can be significantly better. However, if a procedure is used only once, there is no performance gain made by specifying NOUNLOAD. In fact, with NOUNLOAD, a job that uses many different procedures (or windows) can fill up even a large memory space, forcing excessive page faulting and thus degrading performance. In large performance jobs, alternating between UNLOAD and NOUNLOAD can increase performance in other sections. Consider the following example:

```
options unload;
proc print;
proc sort;
proc contents;
proc datasets;
proc access;
run;

options nounload;
proc means data=x;
proc freq data=x;
proc means data=y;
proc freq data=y;
proc means data=z;
proc freq data=z;
proc means data=w;
proc freq data=w;
run;

options unload;
proc dbload;
proc fsedit;
proc sql;
run;
```

Using NOUNLOAD keeps the MEANS and FREQ procedures in memory during the time span when they are used again and again. Once you finish with these procedures, specify UNLOAD to clear memory for upcoming steps.

For more information about the UNLOAD system option, see “UNLOAD” on page 448.

## Enabling Privileged Unloading

The SAS System uses dynamic loading. Dynamic loading allows a module to be called that was not originally linked to the base image. Dynamic unloading disassociates a called module from the base image. Dynamic loading and unloading together reduce resource usage and modularize the SAS System.

Resource usage is reduced because resources are only allocated when they are going to be used. Modularization helps develop and maintain the software, while minimization of resources enhances operating system efficiency. Although OpenVMS does not support a method of completely dynamically unloading images, the SAS System frees certain resources allocated by an image. For example, virtual address space can be deleted and channels freed. There are also resources that require privilege

to free. These resources are allocated out of your process control region. The privileged unloader dynamically frees these resources back to the operating environment.

To minimize page faulting, the base image size has been kept to a minimum using modularization. Additional products can easily be added because individual modules can be sent to users without reinstalling the entire SAS System. You will notice performance gains because pages are faulted into a working set only when they are needed. For instructions about how to install and enable privileged unloading, refer to *Installation Instructions and System Manager's Guide for the SAS System under OpenVMS Alpha and VAX*.

You can use the UNLOAD system option at any time in a SAS session, but the new specification takes effect only after a DATA step or procedure (SAS task). Images can also be automatically removed during resource-critical situations. The SAS System prompts you in these situations for which resources to free and which to keep. In extreme situations, the SAS System takes whatever steps are necessary to continue.

## Disabling Privileged Unloading

For instructions about how to de-install and disable privileged unloading, refer to *Installation Instructions and System Manager's Guide for the SAS System under OpenVMS Alpha and VAX*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for the OpenVMS Environment, Version 8,* Cary, NC: SAS Institute Inc., 1999. 518 pp.

**SAS® Companion for the OpenVMS Environment, Version 8**



1–58025–526–4



1st printing, October 1999



The Institute is a private company devoted to the support and further development of its software and related services.

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.