Paper 300

Data Libraries on Tape

Curtis A. Smith, *Defense Contract Audit Agency, La Mirada, Ca*

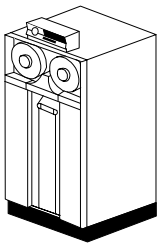
ABSTRACT

Big data forces the SAS System® developer and user to turn to big storage media. Often this means - *tape*. While tape solves the storage space dilemma, it comes at a price. There are limitations, problems, and inefficiencies inherent in tape storage. Within the pages to follow, the author will explore many such difficulties with tape and the ways to overcome the difficulties. For example, tape devices are sequential storage units rather than random access devices, as are disks. What problems does sequential storage bring? Can multiple SAS data sets be stored within a single SAS data library on tape without destroying each other? What effective SAS capabilities are sacrificed when using tape? How does the user allocate a SAS data library on tape from within SAS? All these questions and more will be answered in the pages to follow as the reader learns how to use SAS data libraries on tape.

INTRODUCTION

When our SAS data sets get too big to store on disk devices, all is not lost. The SAS System can use tape devices as SAS data libraries almost as well as it does disk devices. When your SAS data libraries are on tape, there are some things you cannot do that you can do SAS data libraries on disk. Most of what you will do will not be any different when using tape devices. A few things must be overcome, and I have a few tips I have learned after years of using SAS data libraries on tape. Most of the discussion draws from MVS®, but other operating system will apply to the techniques. We will look at the differences between SAS data libraries on tape and those on disk; how to allocate new and existing SAS data libraries on tape; limitations with SAS when using SAS data sets stored in tape libraries; and my favorite tape library tips.

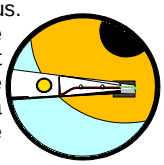
TAPE VS. DISK LIBRARIES



SAS data libraries on tape and SAS data sets stored within them function like those on disk, with a few exceptions. However, the physical structure at the operating system level differs noticeably. Before we look at the functional differences, we will examine the physical differences.

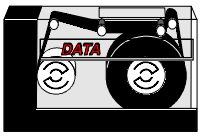
ACCESS METHOD

Tape devices store data sequentially. They store all the bytes one after another. To get to one observation the tape device must read all the observations stored before it - to get to one byte, the tape device must read all the bytes stored before it. Before the SAS DATA step or procedure can complete, it must read the entire file (unless limited by the OBS= option) and maybe the entire SAS data library. So, when using tape devices, we must read meticulously through the tape, as we cannot access the device randomly. Because SAS and the operating system must read the tape sequentially, some limitations are imposed upon us. There are some things in the SAS DATA step we cannot do, and some SAS procedures that will not work. We will discuss solutions to many of the limitations. The good news is, however, that like a SAS data library on disk, you can store multiple SAS data sets within the SAS data set on tape.



PHYSICAL ATTRIBUTES

Our SAS data libraries on tape (tape library) have physical attributes that differ from those of our SAS data libraries on disk (disk library). Under MVS, our DD parameters are different. While we do not use the SPACE parameter, we do need to set the data control block (DCB) parameters differently than we do for a disk library. The record format (RECFM) and the logical record length (LRECL) can be set to a variety of values to optimize performance. Typically, the block size (BLKSIZE) will optimally equal "32760." The SAS Companion for your host environment has the physical attributes for all the parameters you might need to set. Michael A. Raithel discusses the physical attributes of SAS data libraries on tape very thoroughly in his book "Tuning SAS Applications in the MVS Environment."



When we allocate a disk library, we can specify how much space we want using the SPACE parameter. In contrast, with tape devices, we do not use the SPACE parameter; SAS and the operating system will span multiple reels and cartridges. (The SAS Institute has a Usage Note that this does not work with 9-Track tapes.) However, your site might have a default number of maximum tapes to span. This will limit the number of tape units you can span. To increase that number, use the JCL® VOLUME parameter in the DD statement when you create the tape library to tell the operating system how many more tapes that can be used beyond the default. For example, the parameter

```
VOLUME=(,,,10)
```

says to use up to 10 more tapes after the default.

Another difference that is important is device mounts. Unlike SAS data libraries on disk, which can readily access our many SAS data libraries at once, when our SAS data libraries are on tape, we need device mounts for each tape we want. As we will see later, this may create problems for our SAS job. We may find it useful to know the number of tape devices our site has available and the number of tape mount requests made at any one time.

ASSIGNING TAPE DATA LIBRARY

How we allocate *existing* tape libraries differs very slightly from allocating disk libraries. However, how we allocate *new* tape libraries differs drastically. If not done right will result in much lost effort.

ALLOCATING A NEW TAPE DATA LIBRARY

New SAS data libraries on tape under MVS must be allocated externally from SAS. SAS will let you issue a LIBNAME statement while using a DISP= parameter of NEW with a tape unit request. However, the results will be unpredictable. If you do allocate a tape library internally and put one or more SAS data sets in the tape library, your SAS job will appear to end normally. When you try to use the tape library later with another job, you might see the following error:

```
SVC99 error rc=4, reason=1708 : IKJ56228I DATA SET x
NOT IN CATALOG OR CATALOG CAN NOT BE ACCESSED.
```

You should create a tape library with JCL or a TSO® ALLOCATE statement. A typical example of the JCL to create a tape library would be:

```
//TAPELIB DD DSN=HILEVEL.SAS.TAPE.AB1997,
//           DISP=(NEW,CATLG,DELETE),
//           DCB=(RECFM=U,LRECL=32756,BLKSIZE=32760),
//           VOLUME=(,,,10),UNIT=TAPE
```

When you create your new tape libraries, I strongly recommend that you catalog them. Assigning existing SAS data libraries is much simpler if the physical file is cataloged. In fact, SAS cannot allocate a library with a LIBNAME statement if the physical file is not cataloged.

ALLOCATING AN EXISTING TAPE DATA LIBRARY

You can allocate an existing tape library under MVS with JCL, a TSO ALLOCATE statement, or a SAS LIBNAME statement (provided the physical file is cataloged). When using a LIBNAME statement you can add the engine parameter, TAPE, after the libref and before the physical file name. This will help SAS find the physical file. If you fail to specify the TAPE engine on the LIBNAME statement, SAS will issue two tape mount requests. This is because SAS must open the library to figure out what type of engine to assign, causing the first mount request. Then, when the library is opened for read or write, another tape mount is requested. This, obviously, wastes time. To prevent this, specify the TAPE engine on the LIBNAME statement.

When you allocate a file under MVS in a batch job using JCL, the job will stay on the input queue until the files are allocated. If your SAS data library is on tape, the job will stay on the input queue until the needed tape can be mounted. On the other hand, when you allocate your tape library from within SAS using a LIBNAME statement, the job will start without the tape mounted. When the LIBNAME statement is encountered, the operating system will try to mount the tape. If the tape does not mount right away, SAS will abnormally end (ABEND). To prevent this bad thing from happening, you can add the WAIT= parameter to the end of your LIBNAME statement. This will tell SAS to wait the specified number of clock minutes while the system tries to mount the tape. Here is an example.

```
libname tapelib tape 'hilevel.sas.tape.ab1997'
```

If the tape fails to mount within the time you specify, SAS will ABEND. You may find occasions to allocate your SAS data libraries on tape externally, as some host allocation parameters are not available to the SAS LIBNAME statement.

Some installations restrict you from allocating your tape libraries from SAS interactively. The SAS Display Manager System ® supports allocating SAS data libraries on tape, but your site may not allow it. If your site does not support tape libraries interactively, you will likely see the following message in your log:

```
SVC99 error rc=4, reason=0218 : IKJ56221I DATA SET x
NOT ALLOCATED, VOLUME NOT AVAILABLE+IKJ56221I VOLUME
NECESSARY TO SATISFY YOUR REQUEST NOT ON SYSTEM, AND
CANNOT BE MOUNTED.
```

Apparently, some site administrators assume that if your data is so big it needs to be on tape then you should be working in batch mode.

TAPE LIBRARY LIMITATIONS

Unfortunately, not everything that we can do with SAS data sets and SAS data libraries on disk work when our SAS data libraries are on tape. Some aspects of the SAS System just will not work when processing SAS data sets on tape. There are other things that we might be able to do, but we will need to take some precautions or make some detours.

STUFF YOU JUST CANNOT DO

I have found that there are some aspects of the SAS System that you just cannot do: indexing, compressing, and the DATASETS procedure. This is very unfortunate. Indexing is very useful when sorting, summarizing, merging, and WHERE selecting. However, the way indexing operates is incompatible with sequential reading. So, it does not work. Neither does SAS data set compression. This is not such a big deal, however. When storing SAS data sets on tape, we can span as many tapes as we need and tapes are not very expensive. So, compression is not so critical for SAS data sets on tape libraries.

Being unable to use PROC DATASETS is also very unfortunate. If you try to use PROC DATASETS, you will get the following error message in your SAS log:

```
Proc DATASETS is not able to process a sequential
library.
```

Being unable to use PROC DATASETS means that you cannot change the name of a SAS data set, delete a SAS data set from a tape library, rename a variable, or add a label to an existing SAS data set or SAS data set variable. You cannot append SAS data sets in any way. This is because appending requires 'update' access, which is not available to sequential libraries. Still, you can copy SAS data sets and get the contents of your SAS data sets - you just cannot do it with PROC DATASETS. You must use the COPY procedure and the CONTENTS procedure. Be sure to label your SAS data sets on tape and their variables when you create them, because you cannot do it later with PROC DATASETS.

PROCESSING LIMITATIONS

When you open multiple SAS data sets on the same SAS data library stored on disk, SAS can access each of the SAS data sets simultaneously. SAS can do so because it can access the SAS data sets randomly. If, for example, you have a SAS DATA step and SET two SAS data sets, SAS will allocate both SAS data sets as the DATA step begins. This will work if the SAS data library is on disk. However, if the SAS data library is on tape, this will not work. If you will try, you will get this error message:

```
Attempt to open two sequential members in the same
sequential library.  File x cannot be opened.
```

Remember, the SAS System must read the tape to get to the SAS data set(s) you request. It cannot get to two places on the tape simultaneously. So, within a single DATA step or procedure you cannot reference more than one SAS data set from the same SAS data library. You will need to devise a work-around. Let us say, you want to create a new SAS data set on disk by subsetting two SAS data sets, both of which are stored in the same SAS data library stored on tape. You can set up two DATA steps, creating two subset SAS data sets, one from each of the two SAS data sets on tape. Then, append them with the APPEND procedure.

Sequential SAS data libraries do not store observation information in the header of the SAS data library. This means that SAS does not know how many observations are in a SAS data set stored on tape. This can create some problems. When you sort a SAS data set (SORT, MEANS, SUMMARY procedures, etc.), SAS will allocate the sort work data sets it needs, determine their size, pass the SIZE parameter to the host sort utility, and choose the sort program (unless these options are overridden with an OPTIONS statement). SAS chooses the values for these options based upon the number of observations in the SAS data set. However, when the SAS data set is on tape, SAS does not know the number of observations. Thus, SAS cannot properly set these parameters. It is very easy for a sort routine to fail for lack of sort space when your SAS data sets are stored on tape. SAS will return the following message:

```
Sort did not complete successfully.  See messages on
the Job Console Log or //SYSOUT data set.
```

To overcome this, specify the following sort options with an OPTIONS statement.

SORTPGM=	use this option to tell SAS to use the host sort utility or the SAS sort program. Typically, the host sort utility, such as DFSORT, SYNCSORT, or DJSORT, will do better on large SAS data sets. Select either 'SAS' for the SAS sort utility or 'HOST' for the host sort utility.
SORTWKNO=	use this parameter to tell SAS how many sort work data sets to use. This is a numeric value from 1-6.
SORT=	use this parameter to specify the minimum size of all sort work data sets. This is a numeric value for the number of cylinders. SAS will divide the size you select by the number of sort work data sets to figure out the size of each sort work data set.
SORTSIZE=	use this to pass a value to the host sort utility for its SIZE parameter. This is a numeric value.

Here is an example:

```
options  sortpgm=host  sortsize=256k  sortwkno=6
         sort=2400;
```

This will cause SAS to use the host sort utility, with 256K for its SIZE parameter, and allocate six sort work data sets of 400 cylinders each. How much space you need for sorting will be a bit larger than the SAS data set would occupy if it were stored on disk. You may need trial and error to find the right values to use.

STRANGE THINGS

SAS is a fantastic tool. It will let you do just about anything. Nonetheless, in my travels, I have encountered some strange stuff. Here are a few wild ones.

- When you create a tape library, if you do not put a SAS file in it during the same SAS session, the SAS data library will be unusable later. If you create a SAS data library on tape, with JCL for example, and do not put a SAS file in it, the next time you try to use the tape library you might get the following error:

```
SVC99 error rc=4, reason=1708 : IKJ56228I DATA SET x
NOT IN CATALOG OR CATALOG CAN NOT BE ACCESSED.
```

  The solution is to put something in the tape library when you create it. You can create an empty SAS data set, using a DATA step, INPUT statement, and CARDS statement. You do not have to put any observations in it. Just so there is a SAS file on the tape. Alternatively, you can use PROC COPY to copy a SAS data set or an empty SAS file to the tape.

- While you can store multiple SAS data sets in the same tape library, you must be careful when modifying any one of them. When you store SAS data sets on disk and add to one of them, the SAS System can add to the file by placing the new data in the SAS data library it can find free space. This is not true when the SAS data library is on tape. The SAS System will store all the observations of a SAS data set sequentially. This means that if you add to a SAS data set, the SAS System will add the new observations to the end of that SAS data set. What about the other SAS data set stored right after the one you just modified? The SAS System will not move other SAS data sets further down the tape. Because tape is a sequential device that just does not work. The SAS System will destroy all of the subsequent SAS files on the tape. You will get the following error the next time you try to use the other files:

```
SYSTEM ABEND 004 OCCURRED OUTSIDE OF THE SAS
ENVIRONMENT
```

  This error occurs outside the SAS System environment because the SAS data library has been corrupted and cannot be read by the SAS System. If you want to modify a SAS data set stored on tape and you have other SAS data sets stored on that tape, you must first copy the others out of the way and then copy them back after modifying the target SAS data set. You can do this by creating a temporary tape or disk library, use PROC COPY to copy the other SAS data sets to the temporary library, modify the target SAS data set, then use PROC COPY to copy the other SAS data sets back to the tape.

- SAS has documented that under CMS ®, if you use PROC COPY to add a new SAS data set to a tape library, SAS will overwrite the entire SAS data library. The work-around is to use PROC COPY to copy all the SAS data sets from the tape to a disk library where the new SAS data set is also stored. Then use PROC COPY to copy the entire library to the tape.

- At some site under MVS, the expiration date of '99365' (Julian date) given to a physical file means 'no expiration'. However, at some sites, specifying this date on the JCL tape label causes the tape be placed in a 'hold' status that allows the owner to read but not write to the tape. Thus, a SAS data library created on tape may be accessible for writing when it is created, but then may become unavailable for writing. When trying to write to the tape later, the following error message will be contained in the SAS log:

```
SYSTEM ABEND 004 OCCURRED OUTSIDE OF THE SAS
ENVIRONMENT
```

  This error occurs outside the SAS environment because the operating system is not allowing the tape to be modified. Therefore, SAS reports the problem to be something other than itself. To overcome this situation, specify a different date for the expiration date.

OTHER TIPS

Here are a few ideas to help make your tape experience a bit better.

USE DISK ANY WAY

Put your large SAS data libraries on disk, rather than tape, if you only use the SAS data libraries infrequently and your site automatically archives its disk devices. If you do this, after a week or so (depending upon your site) the unused SAS data libraries on disk will archive to tape. There, you will receive the storage costs of tape. When you allocate the archived library, the operating system will retrieve the SAS data library and put it back on disk. This method allows you to use all the features of SAS, while spending just a bit more in storage while the SAS data libraries are on disk. This works well at my site where we have dozens of SAS data libraries, but need only one or two for a project that may last a while, and then do not use them again for a long time. We seldom have more than a couple on disk at a time.

ORDER OF PLACEMENT ON TAPE

When you store multiple SAS data sets in the same tape library, optimize performance by the order in which you place the SAS data sets. Remember, when you need one of the SAS data sets, SAS must read all of the tape until it gets to the SAS data set you want. Unless all of the SAS data sets are of equal size and you use all of them equally as often, the order you place them can affect performance. Consider an example of two SAS data sets on the same tape library, one file is big, the other is huge. The big SAS data set you need daily, the huge SAS data set you need weekly. You place the huge SAS data set on the tape first. Each time you use the big SAS data set, which you do daily, the SAS System must first read the huge SAS data set. In this example, you would improve performance by placing the big SAS data set first. Then you will not have to read vast amounts of tape to get to the big SAS

data set and when you need the huge SAS data set once each week, you will only have to read passed the big SAS data set.

KEEP JUST WHAT YOU NEED

SAS data sets stored in tape libraries must be read sequentially, observation by observation, byte by byte. The more observations there are, the longer it will take to get to the end. The more variables in each observation, the longer it will take to get to the next observation and the longer it will take to get to the end. So, do not store variables and observations you do not need. Coordinate carefully with your developers and users to decide what you do and do not need to keep. Also, redundant values, like descriptions, can be kept in tables and retrieved when necessary. Consider an observation of 150 characters having an account description variable of 30 characters. You can reduce the size of the entire SAS data set by 20% by deleting the description variable and keeping the values in a look up table.

ELIMINATE THE NEED FOR MULTIPLE TAPE DEVICES

Your site has a finite number of tape devices. When you need multiple tape libraries in the same SAS job, if you are not careful and plan, your job will require the operating system to allocate as many tape drives as you specify tape libraries. You may be waiting awhile for the tape drives to become available. Consider the example of reading a huge SAS data set into a SAS DATA step where you want to split the file into five new SAS Data Sets. Each of the SAS data sets will be stored in a separate tape library. You specify each libref and SAS data set name on the DATA step like this:

```
data tape1.file tape2.file tape3.file tape4.file tape5.file;
  set tapein.input;
  more SAS statements
run;
```

Here, you will need six tape devices at once. Your job will wait, maybe for hours, for six drives to become available. Then, when your job grabs six tape drives, everyone at your site will wait for your job to release the drives. Not a very good plan. Instead, you can create the subsets one at a time. This will require only two tapes at once. You can write your code with as a SAS macro like this:

```
%macro subset;
data &lib..file;
  set tapein.input;
  more SAS statements
run;
%mend;
%let lib=tape1;
%subset;
%let lib=tape2;
```

To keep the operating system from still allocating all your tapes at once, use the UNIT=AFF parameter in your JCL or SAS LIBNAME statement on the second through the last tape library to which you are writing. The JCL would look like this:

```
//TAPEIN DD DSN=HILEVEL.SAS.TAPEIN,DISP=OLD
//TAPE1  DD DSN=HILEVEL.SAS.TAPE1,DISP=OLD
//TAPE2  DD DSN=HILEVEL.SAS.TAPE2,DISP=OLD,
          UNIT=AFF=TAPE1
//TAPE3  DD DSN=HILEVEL.SAS.TAPE3,DISP=OLD,
          UNIT=AFF=TAPE2
//TAPE4  DD DSN=HILEVEL.SAS.TAPE4,DISP=OLD,
          UNIT=AFF=TAPE3
//TAPE5  DD DSN=HILEVEL.SAS.TAPE5,DISP=OLD,
          UNIT=AFF=TAPE4
```

The UNIT AFFINITY option will tell the operating system to load the tape on the same unit as it loaded the referenced tape. In our example, we will need two tape devices, one for TAPEIN and one for TAPE1 through TAPE5, reading them one tape at a time.

This example may be a good use of tape devices, but it is a poor use of processing, because we must read the huge source SAS data set five times. Instead, we could use our original DATA step idea, but use tape1 for the first of our five output SAS data sets and put the others in temporary disk libraries. Then, we could use PROC COPY to copy the four temporary SAS data sets to our four other tape libraries, one at a time. Or, we could use five temporary libraries, eliminating the need for more than one tape device. If we do that we will need to alter our UNIT=AFF a bit. Yet whether we use four or all five temporary libraries, we still use the UNIT AFFINITY to limit the number of tape devices we need.

CONCLUSION

SAS data libraries on tape provide most of the functionality of SAS data libraries on disk and offer something that disk libraries do not have - practically an unlimited amount of storage space. Careful planning and execution will help you overcome those few limitations inherent in tape libraries. But do not let those limitations turn you away from using tape.

REFERENCES

PROC DATASETS:
SAS Procedures Guide, Version 6, Third Edition, Chapter 17

SAS MACRO Variables:
SAS Guide to Macro Processing, Version 6, Second Edition, Chapter 2
SAS Language Reference, Version 6, First Edition, Chapter 20

BLKSIZE
SAS Companion for the MVS Environment, First Edition, Chapter 17
Tuning SAS Applications in the MVS Environment, Michael A. Raithel, Chapter 4

LIBNAME Statement:
SAS Language Reference, Version 6, First Edition, Chapter 9

DATA Step:
SAS Language Reference, Version 6, First Edition, Chapter 2

SORT Options:
SAS Companion for the MVS Environment, First Edition, Chapter 17

Usage Notes
V6-LIBNAME-6768, V6-SYS.SYS-2429, V6-APPEND-3874, V6-COPY-8977

Questions: e-mail the author at casmith@mindspring.com