CHAPTER 5 Using SAS Files

Introduction

This section discusses the concept of a SAS data library in the OpenVMS operating environment, sharing data between OpenVMS platforms, and accessing SAS files (including assigning librefs and logical names) on tape.

SAS Data Libraries

An OpenVMS directory can contain many different types of files, including SAS files. All SAS files in a directory that are accessed by the same engine belong to a *SAS data library*. Thus, under OpenVMS, a SAS data library is a logical concept rather than a physical one.

Any OpenVMS directory can become a SAS data library when SAS files are stored in that directory; a single OpenVMS directory can contain several SAS data libraries. (See “Multiple SAS Data Libraries in a Single Directory” on page 126.) Also, under OpenVMS, several directories can constitute a single SAS data library if a search-string logical name is assigned to the series of directories. In Version 8, you can have concatenated libraries using a LIBNAME statement or LIBNAME function. (See “Using a Search-String Logical Name to Concatenate SAS Data Libraries” on page 128.)

Accessing SAS Files

In order to access an individual SAS file in Version 6 of the SAS System, you had to first assign a libref or an OpenVMS logical name to the SAS data library. You could then refer to individual SAS files as *libref.member* (or *logical-name.member*), where *member* is the filename of the individual SAS file.

In Version 8, you can still use librefs or logical names as a convenient way of referring to a SAS data library in SAS programs. However, you can also fully specify individual SAS files in most SAS statements and procedures that access SAS files. If portable SAS code is an issue, then using librefs is the recommended method.

Advantages of Using Librefs Rather than OpenVMS Logical Names

Although you can use an OpenVMS logical name to identify a SAS data library to the SAS System, you may want to use a SAS libref instead for the following reasons:

- You cannot assign an engine nor specify any engine/host options with the DCL DEFINE command. SAS uses the procedure described in “How SAS Assigns an Engine When No Engine Is Specified” on page 130 to determine which engine to use. However, it is more efficient to specify an engine explicitly in a LIBNAME statement. Also, the following SAS engines must be specified in a LIBNAME statement because they are not assigned by default: XPORT, SPSS, OSIRIS, and REMOTE.
- OpenVMS logical names are not included in the list that is produced by the LIBNAME LIST statement until after they have been used as librefs in your SAS session. (See “Listing Your Current Librefs” on page 131.)

Assigning Librefs

You can use any of the following methods to assign a SAS libref:

- the LIBNAME statement
- the LIBNAME function
- the LIBASSIGN command
- the SAS Explorer window.

A libref assignment remains in effect for the duration of the SAS job, session, or process unless you either clear the libref or use the same libref in another LIBNAME statement or LIBNAME function.

If you assign a libref from a SAS process, that libref is valid only within that SAS process.

If you clear a libref from within a SAS process, that libref is not cleared from other SAS processes. For information about clearing librefs, see "Clearing Librefs" on page 130.

Using the LIBNAME Statement

The LIBNAME statement identifies a SAS data library to the SAS System, associates an engine with the library, allows you to specify options for the library, and assigns a libref to it. For details about LIBNAME statement syntax, see "LIBNAME" on page 378.

Using the LIBNAME Function

The LIBNAME function takes the same arguments and options as the LIBNAME statement. For more information about the LIBNAME function, see "LIBNAME" on page 302.

Using the LIBASSIGN Command

Perform the following steps to assign a libref using the LIBASSIGN command:

1. Issue the LIBASSIGN command in the command window. The New Library dialog box opens.
2. Specify the libref in the **`Name:`** field.
3. Specify an engine for the libref in the **`Engine:`** field by selecting the default engine or another engine from the drop-down menu. Depending on the engine that you specify, the fields in the **`Library Information:`** area may change.
4. Click on the **`Enable at startup`** box to assign this libref when you invoke SAS.
5. Specify the necessary information for the desired SAS data library in the **`Library Information:`** area. Depending on the engine selected, there may or may not be a **`Path:`**field available for input.
6. Specify LIBNAME options in the **`Options:`** field. These options can be specific to your host or engine, including options that are specific to a SAS engine that accesses another software vendor's relational database system.
7. Select $\boxed{\text{OK}}$.

Using the SAS Explorer Window

Perform the following steps to assign a libref from the SAS Explorer window:

1. From the **`File`** pull-down menu, select **`New`** when the Libraries node in the tree structure is active. The New dialog box opens.
2. Select **`Library`**, and then select $\boxed{\text{OK}}$. The New Library dialog box opens.
3. Fill in the fields in the New Library dialog box, described in "Using the LIBASSIGN Command" on page 125.
4. Select $\boxed{\text{OK}}$.

Multiple SAS Data Libraries in a Single Directory

A SAS data library consists of all the SAS files in the same OpenVMS directory (or in a group of directories—see "Using a Search-String Logical Name to Concatenate SAS Data Libraries" on page 128) that are accessed by the same engine. If a directory contains SAS files that are accessed by different engines, then you have more than one SAS data library in the directory, and you should therefore have a different libref for each engine-directory combination. (You cannot assign the same libref to more than one engine-directory combination. The second assignment merely overrides the first assignment.)

For example, suppose that the directory [MYDIR] contains SAS files that were created by the V8 engine as well as SAS files that were created by the CONCUR engine. You could use the following LIBNAME statements to assign different librefs to the two engines:

```
libname one v8 '[mydir]';
libname two concur '[mydir]';
```

Data sets that are subsequently referenced by the libref ONE are created and accessed using the V8 engine. Data sets that are referenced by the libref TWO are created and accessed using the CONCUR engine. You can then concatenate librefs ONE and TWO and access all files:

```
libname concat (one two);
```

Multiple Librefs for a Single SAS Data Library

You can assign multiple librefs to the same SAS data library (or engine-directory combination), and you can use those librefs interchangeably. For example, suppose that in two different programs you used different librefs for the same data sets. Later you develop a new program from parts of the two old programs, or you use the %INCLUDE statement to include two different programs. In the new program, you could simply assign the two original librefs to each data library and proceed.

The following LIBNAME statements assign the librefs MYLIB and INLIB to the same SAS data library:

```
libname mylib v8 '[mydir.datasets]';
libname inlib v8 '[mydir.datasets]';
```

Because the engine names and SAS data library specifications are the same, the librefs MYLIB and INLIB are identical and interchangeable.

Assigning OpenVMS Logical Names

There are some advantages to using the LIBNAME statement to identify your SAS data libraries to the SAS System. (See "Advantages of Using Librefs Rather than OpenVMS Logical Names" on page 124.) However, you can also use an OpenVMS logical name for the same purpose. To assign an OpenVMS logical name, use the DCL DEFINE command.

Note: Because you cannot specify an engine name in the DCL DEFINE command, the SAS System uses the procedure described in "How SAS Assigns an Engine When No Engine Is Specified" on page 130 to determine which engine to use. △

To use an OpenVMS logical name to refer to a SAS data library, you must define the logical name either outside the SAS System or from your SAS session using the SAS X statement. For example, you can assign the OpenVMS logical name MYLIB to the directory [MYDIR] in either of the following ways:

- **$ DEFINE MYLIB [MYDIR]**
- **$ sas8 x 'define mylib [mydir]';**

Using an OpenVMS Logical Name as a Libref

After assigning an OpenVMS logical name to a directory, you can use the logical name in a SAS job in the same way you would use a libref. For example, if you assigned the OpenVMS logical name MYLIB to a SAS data library, you could then use MYLIB as a libref in a SAS data step:

```
data mylib.a;
   set mylib.b;
run;
```

Similarly, you could use the logical name as a libref in a SAS procedure:

```
proc contents data=mylib._all_;
run;
```

Because the OpenVMS logical name is being used as a SAS name, it must follow the SAS naming conventions. For details about SAS naming conventions, see *SAS Language Reference: Concepts*.

The first time an OpenVMS logical name is used in this manner, SAS assigns it as a libref for the SAS data library. The logical name is not listed by the LIBNAME LIST statement until after you have used it in a SAS statement. (See "Listing Your Current Librefs" on page 131.)

Note: OpenVMS logical names that are defined in a subprocess are not recognized by the current SAS session. However, OpenVMS logical names that are defined in the OpenVMS parent process are available for use during the current session. For information about how to use the X statement or the X command to define an OpenVMS logical name in the OpenVMS parent process, see "Issuing DCL Commands during a SAS Session" on page 36. △

Using an OpenVMS Logical Name in the LIBNAME Statement

Because you cannot specify an engine in the DCL DEFINE command, you may want to use the LIBNAME statement to specify an engine for a SAS data library to which you previously assigned an OpenVMS logical name. You can use the logical name in place of the libref in a LIBNAME statement, as in this example, which associates the BASE engine with the logical name MAIL:

```
libname mail base;
```

Alternatively, if you specify the logical name in place of the *SAS-data-library* argument of the LIBNAME statement, then you can associate both a libref and an engine with the logical name. The following example associates the libref IN and the BASE engine with the data library that is referred to by the logical name MAIL:

```
libname in base 'mail';
```

You can also use the LIBNAME statement to specify portable library options or engine/host options for a SAS data library to which you previously assigned an

OpenVMS logical name. The following LIBNAME statement associates the libref MAIL and the V6TAPE engine with a path that includes the logical name MYDISK. It also specifies the portable library option ACCESS=:

```
libname mail v6tape 'mydisk:[mylib]'
   access=readonly;
```

Using a Search-String Logical Name to Concatenate SAS Data Libraries

If you have several directories that you want to use as a single SAS data library, you can define an OpenVMS search-string logical name to the list of libraries, and then use that logical name in your SAS programs. The list of libraries can include both directories and other logicals. For example, the following X statement assigns the search-string logical name MYSEARCH to the directories [DIR1], [DIR2], [DIR3], and MYLIB2:

```
x 'define mysearch [dir1],[dir2],[dir3],mylib2';
```

When you reference the data set MYSEARCH.TEST1, the SAS System searches [DIR1], [DIR2], [DIR3], and then the directory pointed to by MYLIB2 for the TEST1 data set:

```
data new;
   set mysearch.test1;
   if total>10;
run;
```

You could also use a LIBNAME statement to assign the libref INLIBS to this series of directories. You use the search-string logical name as the *SAS-data-library* specification:

```
libname inlibs 'mysearch';
```

Files that are opened for input or update are opened from the first directory in which they are found. Files that are created or opened for output are always created in the first directory in the search list. For example, if a filename that you specify exists in both [DIR1] and [DIR3], SAS opens the file that is in [DIR1].

From the SAS Explorer's New Library dialog box, you can also specify a search-string logical name to assign a libref. To do this, type the search-string logical name in the **`Path:`** field.

For additional examples of how SAS files in concatenated SAS data libraries are accessed, see "Accessing Files in Concatenated SAS Data Libraries" on page 129.

For more information about search-string logical names, refer to *OpenVMS User's Manual*.

Concealed Logical Names

By default, SAS translates concealed logical names to their full physical specifications when they are used in LIBNAME statements. For example, consider the following definition for the logical name MYDISK:

```
$ DEFINE/TRANSLATION=CONCEALED -
_$ MYDISK $1$DUA100:[MYDISK.]
```

SAS translates the MYDISK concealed logical name to its full physical specification, resulting in the following libref definition:

```
1? LIBNAME MYLIB 'MYDISK:[MYDIRECTORY]';
Note: Libref MYLIB was successfully assigned
as follows:
Engine:  V8
Physical Name: $1$DUA100:[MYDISK.MYDIRECTORY]
```

Note: The EXPANDLNM system option controls whether concealed logical names are expanded and displayed. Use the NOEXPANDLNM form of this option if you do not want your concealed logical names to be expanded and displayed. For more information, see "EXPANDLNM" on page 411. △

Accessing Files in Concatenated SAS Data Libraries

The SAS System uses a set of rules to determine the order in which concatenated directories are accessed. The rules differ depending on whether you are opening a SAS file for input, update, or output:

□ When a SAS file is accessed for *input* or *update*, the first file found by that name is the one that is accessed. In the following example, if the data set SPECIES exists in both the [MYDIR] and **[mydir.datasets]** directories, the one in the **[mydir]** directory is printed:

```
x 'define mysearch [mydir],[mydir.datasets]';
libname mylib 'mysearch';
proc print data=mylib.species;
run;
```

The same would be true if you used the FSEDIT procedure to open the SPECIES data set for update.

□ When a SAS file is accessed for *output*, it is always written to the first directory, if that directory exists. If the first directory does not exist, then an error message is displayed and SAS stops processing this step, even if a second directory exists. In the following example, the SAS System writes the SPECIES data set to the first directory, **[mydir]**:

```
x 'define mysearch [mydir], sas$samp:[sasdata]';
libname mylib 'mysearch';
data mylib.species;
   x=1;
   y=2;
run;
```

If a copy of the SPECIES data set exists in the second directory, it is not replaced.

Accessing Data Sets That Have the Same Name

If you create a new SAS data set from a data set that has the same name, the DATA statement uses the output rules and the SET statement uses the input rules. In this example, the SPECIES data set originally exists only in the second directory, **mydisk:[mydir]**.

```
x 'define mysearch sys$disk:[sas],mydisk:[mydir]';
libname test 'mysearch';
```

```
data test.species;
   set test.species;
   if value1='y' then
      value2=3;
run;
```

The DATA statement opens SPECIES for output according to the output rules, which indicate that the SAS System opens a data set in the first of the concatenated directories (**sys$disk:[sas]**).

The SET statement opens the existing SPECIES data set in the second directory(**mydisk:[mydir]**), according to the input rules. Therefore, the original SPECIES data set is not updated. After the DATA step is processed, two SPECIES data sets exist, one in each directory.

How SAS Assigns an Engine When No Engine Is Specified

It is always more efficient to explicitly specify the engine name than to ask the SAS System to determine which engine to use. To assign an engine name, use a LIBNAME statement or LIBNAME function, or the LIBASSIGN command in the command window. If you use the LIBASSIGN command the **Default** engine—that is, BASE—is listed in the **Engine:** field of the New Library dialog box; when you select OK, you automatically select this default engine.

If you do not assign an engine name, SAS looks at the OpenVMS file types of the files that exist in the specified directory and uses the following rules to determine which engine to assign:

- If the directory contains SAS data sets from only one of the native library engines that are supported, then that engine is assigned to the libref.
- If the directory contains no SAS data sets, then the default engine is assigned to the libref. The default engine is determined as follows:
  - For SAS data libraries on disk, the default engine is determined by the value of the ENGINE= system option. By default, the ENGINE= system option is set to V8. However, you can change the value of this system option if you prefer to use a different engine as the default engine for disk libraries. Other valid values are V7, V6, and CONCUR. For more information about the ENGINE= system option, see “ENGINE=” on page 410 and *SAS Language Reference: Dictionary*.
  - For sequential-format SAS data libraries (either on tape or disk), the default engine is determined by the value of the SEQENGINE= system option. By default, SEQENGINE= is set to TAPE. The other valid values are V8TAPE, V7TAPE, and V6TAPE.
- A directory that contains SAS data sets from more than one engine is called a *mixed-mode library*. The SAS System assigns the default engine to mixed-mode libraries.

Clearing Librefs

To disassociate a libref from a SAS data library, use the following forms of the LIBNAME statement or the LIBNAME function, where *libref* is the libref of the data library that you want to clear:

LIBNAME statement:

LIBNAME *libref* <CLEAR>;

LIBNAME function:

LIBNAME(*libref*)

In both cases, the libref is cleared only if there are no open files that are associated with that libref, and the libref is cleared only for the SAS process or job from which you submit the LIBNAME statement or function.

You can also use the SAS Explorer window to clear librefs, as follows:

1 With the tree structure activated, select the libref from the Libraries node.
2 With the cursor on the highlighted libref, click and hold the right mouse button (MB3).
3 A pop-up menu opens.
4 Select **`Delete`**.

Listing Your Current Librefs

As in other operating environments, you can use the following form of the LIBNAME statement under OpenVMS to list the attributes of all the librefs that are assigned for your current SAS process:

LIBNAME _ALL_ LIST;

You can also use the SAS Explorer window to see information about your currently assigned SAS data libraries, as follows:

1 From the tree structure, select **`Libraries`** to list all assigned librefs.
2 Select **`View`** and then select **`Details`** to list attributes of the assigned librefs.

You can also see the information using the Properties dialog box. Select the libref. With the cursor on the highlighted libref, click and hold the right mouse button (MB3). A pop-up menu opens. Select **`Properties`**.

In both cases, OpenVMS logical names that you have assigned to SAS data libraries are also listed, but only after you have used them as librefs in your current SAS process. (See "Using an OpenVMS Logical Name as a Libref" on page 127.)

Estimating the Size of a SAS Data Set

To obtain a rough estimate of how much space you need for a disk-format SAS data set that was created by the V8 engine, follow these steps:

Note: This procedure is valid only for *uncompressed* native SAS data files that were created with the V8 engine. △

1 Use the CONTENTS procedure to determine the size of each observation. (See "Using the CONTENTS Procedure to Determine Page Size" on page 131.)
2 Multiply the size of each observation by the number of observations.
3 Add 10 percent for overhead.

Using the CONTENTS Procedure to Determine Page Size

To determine the length of each observation in a Version 8 SAS data set, you can create a Version 8 SAS data set that contains one observation. Then run the

CONTENTS procedure to determine the observation length. The CONTENTS procedure displays **Engine/Host-Dependent Information**, including page size and the number of observations per page for uncompressed SAS data sets. For example, the following input produces a SAS data set plus PROC CONTENTS output:

```
data oranges;
   input variety $ flavor texture looks;
   total=flavor+texture+looks;
   datalines;
navel 9 8 6
;
proc contents data=oranges;
run;
```

The output is shown in Output 5.1 on page 132.

Output 5.1 CONTENTS Procedure Output

```
                           The CONTENTS Procedure

Data Set Name: WORK.ORANGES                        Observations:         1
Member Type:   DATA                                Variables:            5
Engine:        V8                                  Indexes:              0
Created:       10:54 Friday, May 29, 1999          Observation Length:   40
Last Modified: 10:54 Friday, May 29, 1999          Deleted Observations: 0
Protection:                                        Compressed:           NO
Data Set Type:                                     Sorted:               NO
Label:


             -----Engine/Host Dependent Information-----

  Data Set Page Size:         8192
  Number of Data Set Pages:   1
  First Data Page:            1
  Max Obs per Page:           203
  Obs in First Data Page:     1
  Number of Data Set Repairs: 0
  File Name:                  SASDISK:[SASDEMO.SAS$WORK2040F93A]ORANGES.SAS7BDAT
  Release Created:            8.00.00P
  Host Created:               OpenVMS
  File Size (bytes):          16384


           -----Alphabetic List of Variables and Attributes-----

                    #    Variable    Type    Len    Pos
                    ________________________________
                    2    flavor      Num       8      0
                    4    looks       Num       8     16
                    3    texture     Num       8      8
                    5    total       Num       8     24
                    1    variety     Char      8     32
```

To determine page size, the only values that you need to pay attention to are

Observation Length
: is the record size in bytes.

Compressed
: has the value NO if records are not compressed, and the value YES if records are compressed. (If the records are compressed, do not use the procedure given in "Estimating the Size of a SAS Data Set" on page 131.)

Sharing Data between OpenVMS Platforms

SAS files that were created in an OpenVMS operating environment other than the one on which the user is currently running are described as *nonnative*. For example, data sets that were created on the VAX platform are defined as nonnative when they are moved to an Alpha platform.

Nonnative data must be converted before it can be accessed. There are two ways to convert nonnative data:

- convert the data "transparently" between the OpenVMS VAX format and the OpenVMS Alpha format each time you access the file. This method causes performance degradation.
- convert the data to the local format one time only. This method is more efficient, eliminating the need to convert the data each time you access it.

The following is an example of how you can convert a file:

```
data a;
   set b;
run;
```

This code reads file B, which is in nonnative format, and creates a native version A.

A limitation of the one-time-only conversion is that the OpenVMS VAX platform supports a minimum numeric variable length of 2 bytes. The OpenVMS Alpha platform supports a minimum numeric variable length of 3 bytes. Therefore, using this method to move data from the VAX platform, which supports 2-byte numeric storage, to the Alpha platform, which supports 3-byte numeric storage, is not permitted. Instead, to move data from the VAX platform to an Alpha platform, you must use the VAXTOAXP procedure. Consequently, there is a potential loss of numeric precision when you move data from a VAX platform to an Alpha platform. For more information, see Chapter 9, "Data Representation," on page 199. For more information about the VAXTOAXP procedure, see "VAXTOAXP" on page 350.

Note: In Version 8, the VAXTOAXP procedure increases VAX numeric variables of length two to seven characters by one character to minimize loss of precision. △

Multiuser Access to SAS Files

Under certain circumstances, a SAS file can be accessed by more than one user concurrently. This feature enables different users, or the same user from different processes, to access the same SAS file at the same time without conflict. However, to prevent problems of integrity or data conflict, multiple accesses of data are sometimes blocked. The following rules summarize the conditions for allowing or disallowing multiple access to the same SAS file with any engine except the CONCUR engine:

- □ If a file is open for input, another user may also open that file for input. The same process may also open the file for output, but all other access is denied.
- □ If a file is opened for update, all other access is denied.
- □ If a file is opened for output, the same process may also open the file for input if the file previously existed, but all other access is denied.

The one exception to these rules is when an OpenVMS search-string logical name is used as the physical path of a LIBNAME statement. In this case, when a SAS file is opened for input, another user may open that file for input. If the file is opened for update or output, all other access is denied, including access by the same process.

Under OpenVMS, the concurrency engine (CONCUR) allows concurrent read and write access to native data sets. For details, see "Using the CONCUR Engine" on page 154.

Accessing SAS Files on Tape

This section discusses accessing, reading, and writing SAS files on tape. There are also some notes on tape usage.

DCL Commands for Tape Access

In order to write to a tape in a SAS job, you can issue the following DCL commands to allocate the tape drive and mount the appropriate tape volume. You must issue these commands in the order shown:

```
$ ALLOCATE tape-device:
$ INITIALIZE tape-device:  volume-label
$ MOUNT tape-device:  volume-label
```

Note: If you are writing SAS files to tape with the TAPE engine, you must mount the tape as a labeled Files-11 tape. A labeled Files-11 tape has header information preceding each file. An unlabeled, or foreign, tape does not have this header information. The TAPE engine can process only labeled Files-11 tapes. For more information about Files-11 tapes, refer to *Guide to OpenVMS Files and Devices*. △

CAUTION:

Issue the INITIALIZE command only if you are writing to a tape for the first time. When a tape is initialized, any files that were previously stored on the tape are no longer accessible. Therefore, use the ALLOCATE and MOUNT commands when you want to read from a tape or write additional files to a tape; do not reinitialize the tape. △

The volume label that you specify in the INITIALIZE command must be used subsequently in the MOUNT command in order to access the tape. After you have issued the appropriate commands to access the tape, you must then use the LIBNAME statement to associate a libref with the tape.

When your SAS job finishes, issue the following commands to release the tape drive from your terminal session:

```
$ DISMOUNT tape-device:
$ DEALLOCATE tape-device:
```

Any of these commands can also be issued in the X statement. However, if you use the X statement, you must issue the INITIALIZE command before the ALLOCATE command. The reverse order is not supported when you use the X statement.

Accessing Multivolume Tapes

When creating SAS files on multivolume tapes, you must initialize the tapes *before* you write the files to the tapes. If you do not initialize the tapes first, the operating environment will not recognize them as part of the same volume set. When you mount the first volume of the set, use the following MOUNT command:

$ MOUNT/INITIALIZE=CONTINUATION -

_$ *tape-device*: *label*

This command instructs the OpenVMS system to add a continuation number to each label as it creates the multivolume set. For example, if you have a series of tapes initialized to MYTAPE and use drive MUA0:, use the following command:

```
$ MOUNT/INITIALIZE=CONTINUATION MUA0: MYTAPE
```

When the first volume is filled, the operating environment prompts the operator to mount MYTA02. The OpenVMS system adds a sequencing number to the tape label. As tape labels are limited to six characters, the original label, if it exceeds this number, can be truncated when the continuation number is added.

Reading and Writing SAS Files on Tape

In addition to the appropriate DCL commands, use the LIBNAME statement or the New Library dialog box to associate a libref with the tape when reading or writing SAS files. The following is an example of the LIBNAME statement:

```
libname sample tape 'mua0:';
```

Then use the libref SAMPLE in the appropriate SAS statements to refer to the tape. The following is an example:

```
data sample.oldstat;
   set status;
run;
```

A libref associated with a tape drive signals that the file to be read or written is in sequential format.

Note: You can also write SAS files in sequential format on disk if you define the libref to a disk location, but use the sequential engine (TAPE) in the LIBNAME statement or in the New Library dialog box. △

The tape can contain one or more SAS files. When you read or write a file on tape, you use a two-level name; the first level is a libref that refers to the tape, and the second level names the SAS file to be read or written. The following is an example of the LIBNAME statement:

```
libname mytape 'mua0:';
data diskds1;
   set mytape.ds1;
run;
```

This program reads a data set with the filename DS1.SAS7SDAT from the tape referenced by the libref MYTAPE.

You can write SAS files with duplicate names to the same tape. For example, you can have more than one SAS data set named DS1 on a tape. When you read the data set named DS1, the first (and possibly the oldest) version of DS1 found on the tape is the version read. The first version found depends on the current position of the tape.

Notes on Tape Usage

Use the COPY procedure to copy existing SAS files from disk to tape. The following is an example:

```
libname mydisk '[dir1]';
libname mytape tape 'mua0:';
proc copy in=mydisk out=mytape;
run;
```

This procedure is often simpler to use than the DCL COPY command when moving SAS files to tape. Also, SAS log notes document the files copied. You can also use the DCL DIRECTORY command to list the SAS files on a labeled tape after it has been created.

When you use the DCL COPY command to move sequential format files created on disk to tape, you must create the files with a page size of 512 bytes and mount the tape with a block size of 512 bytes. The following example creates a sequential format data set on disk. It then shows how to copy it to tape and access it from within the SAS System.

As a first step, create the data set on disk using the sequential engine, with a page size of 512 bytes. Use the BUFSIZE= data set option to set the page size:

```
libname seqdisk tape '[dir]';
data seqdisk.a(bufsize=512);
   ...  more DATA step statements ...
run;
```

Now mount the tape with a block size of 512 bytes and copy the file to tape by issuing the following commands:

```
$ MOUNT/BLOCKSIZE=512 MUA0: MYTAPE
$ COPY A.SAS7SDAT MYTAPE:
```

You can now access this data set directly from within the SAS System, as in the following statements:

```
libname seq tape 'mua0:';
proc contents data=seq.a;
run;
```

If you can, it is far more efficient to create the data set on tape within the SAS System, using the TAPE engine. Use the DCL COPY command only when you have no other alternative. The advantage of using the TAPE engine instead of the DCL COPY command is that when you use the TAPE engine, you can use larger page sizes and block sizes. This means that I/O is more efficient because you can process the data in larger chunks.

To convert the data sets currently in disk format to sequential format before using the COPY command to move them to tape, you can use the following steps:

```
libname mydisk '[dir1]';
libname mytape tape '[dir2]';
data mytape.a;
   set mydisk.a;
run;
```

If you store the files on an unlabeled tape, they must be restored to disk before the SAS System can access them.

Generation Data Sets

Generation data sets are not supported for Version 8 of the SAS System in the OpenVMS operating environment. The GENMAX= and GENNUM= data set options described in *SAS Language Reference: Dictionary* are not supported under OpenVMS.

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

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