CHAPTER 17 Procedures

Procedures in the CMS Environment

Portable procedures are documented in the *SAS Language Reference: Dictionary*. Only the procedures that are CMS-specific or that have CMS-specific behavior are documented in this section.

If the SAS procedure is also described in the *SAS Language Reference: Dictionary*, that information is not repeated here. For procedures that are entirely CMS-specific, a full description is provided.

BMDP

Calls any BMDP program to analyze data in a SAS data set

CMS specifics: all

Syntax

PROC BMDP <*options*>;
 VAR *variables*;

BY *variables*;
PARMCARDS;
BMDP control statements;

Details

BMDP is a library of statistical analysis programs originally developed at the UCLA Health Sciences Computing Facility.

You can use the BMDP procedure in SAS programs to

- call a BMDP program to analyze data from a SAS data set
- convert a SAS data set to a BMDP save file.

The BMDP procedure and the BMDP program that it invokes can require a large virtual machine, so you may have to increase your machine's virtual storage. To use the BMDP procedure in a SAS session, invoke SAS wth the statement:

```
sasbmdp8
```

This command invokes an EXEC that sets up the necessary filerefs for BMDP. If this command is not available on your computer system, ask your computing center staff for help in setting it up.

To use the BMDP procedure, first specify the name of the BMDP program that you want to invoke in the PROC BMDP statement. (Note that a MODULE file must be present for the BMDP program that you want to run.) The VAR and BY statements can follow this, but they are optional. Then, specify the PARMCARDS statement and your BMDP control language. SAS prints the BMDP program output as part of the SAS log. You can use the BMDP procedure multiple times in your SAS job.

PROC BMDP Statement

PROC BMDP <*options*>;

CODE=*save-file*

assigns a name to the BMDP save file that the BMDP procedure creates from a SAS data set. The *save-file* argument corresponds to the CODE sentence in the BMDP INPUT paragraph. For example, if you use the following statement, the BMDP INPUT paragraph must contain the sentence CODE='JUDGES'.

```
proc bmdp prog=bmdp3s code=judges;
```

The CODE= option is usually not necessary in the PROC BMDP statement. When the CODE= is not specified, the name of the BMDP save file is the SAS data set name.

CONTENT=CORR | DATA | FREQ | MEAN

lets BMDP know if your SAS data set is a standard SAS data set (CONTENT=DATA) or if it contains a correlation matrix (CORR), variable means (MEAN), or frequency counts (FREQ). You need not specify the CONTENT= option for specially structured SAS data sets created by other SAS procedures. If you omit the CONTENT= option, the data set's TYPE value is used.

Note: BMDP may use a structure for special data sets (for example, a correlation matrix) that is different from the SAS structure. Be sure that the input SAS data set is in the form that BMDP expects. △

DATA=*SAS-data-set*

specifies the SAS data set that you want the BMDP program to process. If you do not specify the DATA= option, the BMDP procedure uses the most recently created SAS data set.

LABEL=*variable*

specifies a character variable whose values are to be used as case labels for BMDP. Only the first four characters of the values are used. The variable for the LABEL= option must also be included in the VAR statement if you use one.

LABEL2=*variable*

specifies a character variable whose values are to be used as second case labels for BMDP. As with the LABEL= option, only the first four characters are used. The variable for the LABEL2= option must also be given in the VAR statement if you use one.

NOMISS

specifies that you want the BMDP procedure to exclude observations with missing values from being passed to the BMDP program.

PROG=BMDP*nn*

specifies the BMDP program that you want to run. If you do not want to run a BMDP program, but just want to convert a SAS data set to a BMDP save file, omit this option and use the UNIT= option. The following PROC BMDP statement runs the BMDP3S program.

```
proc bmdp prog=bmdp3s;
```

Note: The BMDP MODULE file must be present for the program to run. In some versions of BMDP, MODULE files are provided. Otherwise, you need to create a MODULE file. For help on creating a MODULE file, see your system administrator. △

UNIT=*n*

specifies the FORTRAN logical unit number for the BMDP save file that the BMDP procedure creates. The value that you specify for *n* must correspond to the unit value that is specified in the INPUT paragraph of the BMDP control language.

If you omit this option, the *n* argument defaults to 3 and FT03F001 is used as the fileref for the save file. A warning message is also printed, indicating that 3 is being used as the unit number.

VAR Statement

VAR *variables*;

The VAR statement specifies the variables to be used in the BMDP program. When you do not include a VAR statement, the BMDP program uses all the numeric variables in the SAS data set.

BY Statement

BY *variables*;

Use the BY statement with the BMDP procedure to obtain separate analyses on observations in groups defined with the BY variables. When you use a BY statement, ensure that the observations in the input SAS data set are sorted in the order of the variables listed in the BY statement, or that the data set has an appropriate index. See *SAS Language Reference: Dictionary* for details on what constitutes an appropriate index.

If a BY statement is used, it is printed with the BMDP printed output to distinguish the BY-group output. The BMDP output is printed as part of the SAS listing file.

PARMCARDS Statement

PARMCARDS;

The PARMCARDS statement signals that the BMDP control language follows. This is similar for all BMDP programs. See the most current BMDP manual for information on their forms and functions.

The BMDP INPUT paragraph must include UNIT and CODE sentences. Their values must match the UNIT= option and CODE= option values given in the PROC BMDP statement. (If the PROC BMDP statement does not specify a value for the UNIT= option, 3 is used as the unit value in the BMDP statements.) Use the SAS data set name as the CODE value unless you have specified a different name with the CODE= option in the PROC BMDP statement. Omit the VARIABLES paragraph from the BMDP statements since it is not needed when your input is a save file.

How Missing Values Are Handled

Before the BMDP procedure sends data to BMDP, it converts missing SAS values to the standard BMDP missing values. When you use the NOMISS option in the PROC BMDP statement, observations with missing values are excluded from the data set sent to the BMDP program.

Invoking BMDP Programs That Need FORTRAN Routines

Some BMDP programs, such as those for nonlinear regression, may need to invoke the FORTRAN compiler and linkage editor before executing. All BMDP compilation and link editing must be completed before using PROC BMDP.

Example of Creating and Converting a BMDP Save File

The following actions occur in Example Code 17.1 on page 186:

- DATA TEMP creates a SAS data set called TEMP.
- The CONTENTS procedure shows the description information for the data set TEMP.
- PROC BMDP calls the BMDP program BMDP1D to analyze the data set TEMP. Note the BMDP program statements UNIT=3 and CODE='TEMP'. The results are stored in the BMDP save file BOUT. Also note that the word NEW must be in the SAVE paragraph. UNIT=*nn* should refer to the FT*nn* F001 fileref defined in your file definition statement.
- The CONVERT procedure converts the BOUT BMDP save file to a SAS data set named FROMBMDP. The BOUT save file is in UNIT 4, that is, FT04F001.
- The CONTENTS and PRINT procedures show the new SAS data set, FROMBMDP.

Example Code 17.1 PROC BMDP Example

```
data temp;
   input a b c;
   cards;
1 2 3
4 5 6
7 8 9
;

proc contents;
```

```
   title
   'SAS DATA SET TO BE RUN THROUGH BMDP1D';
run;

proc bmdp prog=bmdp1d data=temp unit=3;
   parmcards;
/prob title='SHOW SAS/BMDP INTERFACE'.
/input unit=3. code='TEMP'.
/save code='BOUT'. new. unit=4.
/end
/finish
;

proc convert bmdp=ft04f001 out=frombmdp;run;

proc contents;
   title
'SAS DATA SET CONVERTED FROM BMDP SAVE FILE';
run;

proc print;
run;
```

The output from Example Code 17.1 on page 186 is shown in Output 17.1 on page 187.

Output 17.1 Output of the PROC BMDP Example

```
1             CONTENTS OF SAS DATA SET TO BE RUN THROUGH BMDP1D            1
                                                12:37 Friday, March 15, 1999

                              The CONTENTS Procedure

Data Set Name: WORK.Temp                          Observations:         3
Member Type:   DATA                               Variables:            3
Engine:        V8                                 Indexes:              0
Created:       12:37 Friday, March 15, 1999       Observation Length:   24
Last Modified: 12:37 Friday, March 15, 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:           338
       Obs in First Data Page:     3
       Number of Data Set Repairs: 0
       File Name:                  Temp Work A
       Release Created:            7.0000B2
       Release Last Modified:      7.0000B2
       Host Created:               VM/ESA
       Last Modified by:           BMDP90V8
       Owner Name:



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

                    #    Variable    Type    Len    Pos
                    -----------------------------------
                    1    a           Num       8      0
                    2    b           Num       8      8
                    3    c           Num       8     16
```

REFERENCE

Dixon, W.D., Brown, M.B., Engleman, L., Frane, J.W., Hill, M.A., Jennrich, R.I., and Toporek, J.D., eds. (1981), *BMDP Statistical Software 1981*, Los Angeles: University of California Press.

CIMPORT

Restores a transport file created by the CPORT procedure

CMS specifics: default disk file, tape device

Syntax

PROC CIMPORT <*options*>;

Details

The DISK option is the default for the CIMPORT procedure. Therefore, PROC CIMPORT defaults to reading from a file on disk instead of from a tape. For disk files, if the fileref SASCAT is not assigned, the default file SASCAT DATA * is searched for. If the TAPE option is specified, SASCAT is temporarily assigned to tape device 181 and is used if a tape is attached. If the tape being processed is other than 181, specify the tape device as the filename in the CIMPORT procedure or FILENAME statement. If you want to read a file from tape, you must specify the TAPE option unless you assigned SASCAT to a tape.

When reading files from tapes, you do not have to specify the DCB attributes in a SAS FILENAME statement or CMS FILEDEF command. However, it is recommended that you specify BLKSIZE=8000.

Here is an example of importing from disk a transport file that PROC CPORT created from a SAS data library on another operating environment:

```
libname myfile 'a';

proc cimport library=myfile;
run;
```

Because no INFILE= option is specified, the default disk file SASCAT DATA * is searched for and used.

Here is an example of importing from tape a transport file that PROC CPORT created from a SAS data library on another operating environment:

```
libname newlib 'a';

proc cimport library=newlib infile='tap3' tape;
run;
```

Because TAP3 is specified, SAS looks to the tape on device 183 for the SAS data sets to import. For a complete listing of symbolic names for CMS tape output devices (such as TAP3), see Table 18.1 on page 230.

In the following example, the fourth file on standard label tape attached at 181 is imported to create a Version 8 SAS library:

```
filename x tape 'tap1' label=sl 4;
proc cimport lib=v5 infile=x tape;
run;
```

See Also

- *Moving and Accessing SAS Files across Operating Environments*
- *SAS Procedures Guide*

CONTENTS

Prints descriptions of the contents of one or more files from a SAS data library

CMS specifics: engine/host-dependent information, directory information

Syntax

PROC CONTENTS <*options*>;

Details

While most of the output generated by the CONTENTS procedure is the same across all operating environments, the Engine/Host Dependent Information output is system-dependent. Output 17.2 on page 190 shows the information specific to CMS for the V8 engine generated from the following SAS statements:

```
data oranges;
   input variety $ flavor texture looks;
   cards;
navel 9 8 6
temple 7 7 7
valencia 8 9 9
mandarin 5 7 8
;

proc contents data=oranges;
run;
```

Output 17.2 Engine/Host Dependent Information from PROC CONTENTS for the V8 Engine

```
                        The SAS System                                1

                             The CONTENTS Procedure

   Data Set Name: WORK.ORANGES                        Observations:         4
   Member Type:   DATA                                Variables:            4
   Engine:        V8                                  Indexes:              0
   Created:       14:27 Tuesday, March 5, 1999        Observation Length:   32
   Last Modified: 14:27 Tuesday, March 5, 1999        Deleted Observations: 0
   Protection:                                        Compressed:           NO
   Data Set Type:                                     Sorted:               NO
   Label:

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

         Data Set Page Size:         6144
         Number of Data Set Pages:   1
         First Data Page:            1
         Max Obs per Page:           254
         Obs in First Data Page:     4
         Number of Data Set Repairs: 0
         File Name:                  ORANGES WORK A
         Release Created:            7.0000B1
         Host Created:               VM/ESA
         Owner Name:


              -----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
                      1    variety     Char      8     24
```

This information can help you optimize the storage space for the data set. For an explanation of this information, see “Estimating the Size of a SAS Data Set” on page 50 .

If you use DATA=_all_ and the DIRECTORY option, SAS generates a list of library members similar to the PROC DATASETS directory information in the SAS log.

See Also

- *SAS Language and Procedures: Usage*
- *SAS Procedures Guide*

CONVERT

Converts BMDP, OSIRIS, and SPSS system files to SAS data sets

CMS specifics: all

Syntax

PROC CONVERT <*options*>;

Details

The CONVERT procedure converts BMDP, OSIRIS, and SPSS system files to SAS data sets.

PROC CONVERT produces one output data set, but no printed output. The new data set contains the same information as the input system file; exceptions are noted under "How Variable Names Are Assigned" on page 193.

The procedure converts system files from these packages:

- □ BMDP save files through and including the most recent version of BMDP.
- □ SPSS save files through and including Release 9, along with SPSS-X and the SPSS Portable File Format.
- □ OSIRIS files through and including OSIRIS IV (hierarchical file structures are not supported).

These software packages are products of other organizations. Changes, therefore, can be made that make new system files incompatible with the current version of PROC CONVERT. SAS Institute cannot be responsible for upgrading PROC CONVERT to support changes to the packages listed previously; however, attempts will be made to do so as necessary with each new release of SAS.

PROC CONVERT Statement

PROC CONVERT <*options*>;

For the PROC CONVERT statement, *options* can be from any of those in the options list presented later in this section. Only one of the options specifying a system file (BMDP, OSIRIS, or SPSS) can be included. Usually, only the PROC CONVERT statement is used, although data set attributes can be controlled by specifying the DROP=, KEEP=, or RENAME= data set options with the OUT= option in this procedure. Refer to *SAS Language Reference: Dictionary* for more information about these data set options. You can also use the LABEL and FORMAT statements following the PROC CONVERT statement.

In the following descriptions, *fileref* is the logical name that you associate with a permanent filename by issuing a SAS FILENAME statement. For example, the following SAS statement associates the logical name BMDPFILE with the CMS file BMDP SAVEFILE A. After defining *fileref*, use it in your program to reference the permanent filename.

```
filename bmdpfile 'bmdp savefile a';
```

Instead of a fileref, you can use a libref if you reference the correct engine (BMDP, SPSS, or OSIRIS) in the LIBNAME statement. For example, you can use the following LIBNAME statement to assign a libref to BMDP SAVEFILE A:

```
libname bmdpfile bmdp 'bmdp savefile a';
```

You can use the following options with the PROC CONVERT statement:

BMDP=*fileref* | *libref* <(CODE=*code-id* CONTENT= *content-type*)>
: specifies the logical name of a BMDP save file. By default, the first save file in the data set is converted. If you have more than one save file in the data set, you can use two additional options in parentheses after the logical name. The CODE= option lets you specify the code of the save file that you want, and the CONTENT= option lets you give the save file's content. For example, if a file CODE=JUDGES has CONTENT=DATA, you can use this statement:

```
proc convert bmdp=save(code=judges content=data);
```

DICT=*fileref* | *libref*
: specifies the logical name of a data set containing the dictionary file for the OSIRIS data set. The OSIRIS= option must be specified if you use the DICT= option.

FIRSTOBS=*n*
: gives the number of the observation where the conversion is to begin. This option enables you to skip observations at the beginning of the BMDP, OSIRIS, or SPSS system file.

OBS=*n*
: specifies the number of the last observation to be converted. This option enables you to exclude observations at the end of the file.

OSIRIS=*fileref* | *libref*
: specifies a logical name for a data set containing an OSIRIS file. You must also include the DICT= option, described earlier, when you use the OSIRIS= option.

OUT=*SAS-data-set*
: names the SAS data set created to hold the converted data. If the OUT= option is omitted, SAS still creates a WORK data set and automatically names it DATA*n*, just as if you omitted a data set name in a DATA statement. If it is the first such data set in a job or session, SAS names it DATA1; the second is DATA2, and so on. If the OUT= option is omitted or if you do not specify a two-level name in the OUT= option, the data set converted to SAS format is not permanently saved. See "Handling Space in the WORK Library" on page 21 for more information.

SPSS=*fileref* | *libref*
: specifies a logical name for a data set containing an SPSS file. The SPSS file can be in any of three formats: SPSS Release 9 (or prior), SPSS-X format (whose originating operating environment is OS/390, CMS, or VSE), or Portable File Format from any operating environment.

How Missing Values Are Handled

If a numeric variable in the input data set has no value or a system missing value, PROC CONVERT assigns it a missing value.

How Variable Names Are Assigned

The following sections explain how names are assigned to the SAS variables created by the CONVERT procedure.

CAUTION:
Be sure that the translated names are unique. △

Variable names are translated as indicated in the following sections.

Variable Names in BMDP Output Variable names from the BMDP save file are used in the SAS data set, except that nontrailing blanks and all special characters are converted to underscores in the SAS variable names. The subscript in BMDP variable names, such as x(1), becomes part of the SAS variable name, with the parentheses omitted: X1. Alphabetic BMDP variables become SAS character variables of length 4. Category records from BMDP are not accepted.

Variable Names in OSIRIS output For single-response variables, the V1 through V9999 name becomes the SAS variable name. For multiple-response variables, the suffix R*n* is added to the variable name, when *n* is the response, for example, V25R1 is the first response of the multiple-response variable V25. If the variable after V1000 has 100 or more responses, responses above 99 are eliminated. Numeric variables that OSIRIS stores in character, fixed-point binary, or floating-point binary mode become SAS numeric variables. Alphabetic variables become SAS character variables; any alphabetic variable of length greater than 200 is truncated to 200. The OSIRIS variable description becomes a SAS variable label, and OSIRIS print format information becomes a SAS format.

Variable Names in SPSS output SPSS variable names and variable labels become variable names and labels without change. SPSS alphabetic variables become SAS character variables of length 4. SPSS blank values are converted to SAS missing values. SPSS print formats become SAS formats, and the SPSS default precision of no decimal places becomes part of the variable's formats. The SPSS DOCUMENT data are copied so that the CONTENTS procedure can display them. SPSS value labels are not copied.

See Also

- "The BMDP, SPSS, and OSIRIS Engines" on page 55

COPY

Copies SAS data files

CMS specifics: use of LABEL= option

Copying Members to Sequential Libraries

When copying members to a sequential library, any existing members in the library are normally lost. The copied members replace the existing members. To preserve existing members in the destination sequential library, you can choose to append copied members to the end of the sequential library without checking for duplicate members. To do so, you need to allocate the destination library using the DISP=MOD option in the LIBNAME statement. This LIBNAME option is valid for libraries using the V6TAPE or later engine, and this option applies only to the COPY procedure. When DISP=MOD is asserted on the destination sequential library, SAS does not check for duplicate members.

Appending new members without checking for duplicates means that the destination library can have two or more members with the same name after the copy. Only the first member in the library will be accessed by SAS. To ensure that all members can be accessed by SAS, use the EXCLUDE statement in the COPY procedure or, after the COPY procedure, use the REMOVE or RENAME statements in the DATASETS procedure. For further information, see *SAS Procedures Guide*.

For further information on sequential tape processing, see "Working with SAS Files on Tape" on page 36.

Types of Label Processing

The type of tape label processing that you choose has a fundamental effect on how you use tapes. To specify how you want label processing done for your SAS program, you can use the LABEL= option in the LIBNAME statement. The examples that follow show three common types of label processing:

- LABEL=LABOFF (turn off label processing)
- LABEL=NL (nonlabeled tape processing)
- LABEL=SL (standard labeled tape processing).

The SAS System under CMS follows the defaults for the CMS FILEDEF command, and therefore the default for this option is LABEL=LABOFF. However, to reduce the risk of error, SAS Institute recommends using LABEL=NL for nonlabeled tapes. Always use LABEL=SL for labeled tapes. SAS Institute also recommends avoiding the use of the CMS FILEDEF command with SAS and instead using the LIBNAME statement.

LABEL=LABOFF Processing If you specify the LABEL=LABOFF option in the LIBNAME statement, you assume all responsibility for positioning the tape before reading or writing a file. SAS begins reading or writing to the tape at its current position. Use CMS TAPE commands (TAPE REW, TAPE FSF, TAPE BSF) to position the tape to the correct point. Note that when you issue CMS TAPE FSF or CMS TAPE BSF, your tape is positioned on *just the other side* of the tape mark from where you started. You should first reissue the LIBNAME statement before issuing any CMS TAPE commands that change the position of the tape. Reissuing the LIBNAME statement clears internal data structures that are kept by SAS and that are invalidated by moving the tape. If there is any uncertainty about tape position, issue CMS TAPE REW to start from a known point, the beginning of the tape.

After SAS reads or writes a tape library, the tape is positioned just after the end-of-file tape mark for that tape library. An exception is that if an error occurs during tape I/O, the tape remains positioned at the point of the error, and not past the tape mark. Normal output is terminated with a single tape mark. After your final output, you should issue CMS TAPE WTM to ensure that the tape ends with a double tape mark.

If you write to a tape library and then want to append to the same tape library, you must issue CMS TAPE BSF so that the tape mark that ends the first output is overwritten. Otherwise, this tape mark separates the two outputs into two tape libraries. When you write data to a tape, everything from that point that was formerly on the tape becomes unusable. Therefore, take extreme care in positioning the tape.

To reread a tape library you have just read or written, manually position the tape to the desired location. To do this, reissue the LIBNAME statement, and then issue CMS TAPE commands to position the tape.

Some tape processing examples using LABEL=LABOFF include

- copying a SAS data library to a new tape:

```
    /* label=laboff default   */
    /* insure tape position   */
libname tapeout 'tap1';
CMS tape rew;
    /* copy SAS data library  */
proc copy in=lib1 out=tapeout;
    /*  from lib1 to tape     */
run;
    /* extra tape mark at end */
CMS tape wtm;
```

- copying multiple SAS data libraries to a single SAS data library on a new tape:

```
    /* label=laboff default   */
libname tapeout 'tap1';
    /* insure tape position   */
CMS tape rew;
    /* copy SAS data library  */
```

```
proc copy in=lib1 out=tapeout;
    /*  from lib1 to tape     */
run;
    /* position tape before   */
CMS tape bsf 1;
    /*  end of file mark      */

    /* copy SAS data library  */
proc copy in=lib2 out=tapeout;
    /*  from lib2 to the      */
run;
    /*  same tape library     */

    /* extra tape mark at end */
CMS tape wtm;
```

- □ copying a SAS data library to tape as a second sequential file (following a preexisting file):

```
    /* label=laboff default   */
libname tapeout 'tap1';
    /* insure tape position   */
CMS tape rew;
    /* use FSF N to forward   */
CMS tape fsf 1;
    /*  past N existing files */

    /* copy SAS data library  */
proc copy in=lib2 out=tapeout;
    /*  from lib2 to a        */
run;
    /*  separate tape library */

    /* extra tape mark at end */
CMS tape wtm;
```

- □ appending a SAS data library to an existing SAS data library on a tape:

```
    /* label=laboff default   */
libname tapeout 'tap1';
    /* insure tape position   */
cms tape rew;
    /* use fsf N to forward   */
cms tape fsf 1;
    /*  past N existing files */
cms tape bsf;
    /* position for appending */
    /* append SAS data library*/
proc copy in=lib3 out=tapeout;
    /*  from lib3 to tape     */
run;
    /* extra tape mark at end */
cms tape wtm;
```

LABEL=NL Processing With the LABEL=NL *n* option of the LIBNAME statement, you specify the sequential number of the file on the tape. For example LABEL=NL 3 causes the third sequential file to be opened. You do not need to use the CMS TAPE command to position the tape; when SAS opens the sequential file, CMS automatically rewinds the tape and positions it to the specified file. Because the tape is rewound during positioning, it will be positioned correctly regardless of the prior tape position.

After SAS reads or writes a tape library, the tape is positioned just after the end-of-file tape mark for that tape library. Output is terminated with a double tape mark; you do not need to write an additional tape mark.

If you do output to a tape library and then want to append to the same tape library, simply continue to use the existing libref. Do not reissue the LIBNAME statement. Likewise, once you have read from or written to a tape library, if you then want to reread it, no special steps are required.

If you want to use PROC COPY to append to a tape library that was written in a previous SAS session or for which the libref has been deassigned, you must do something in the current SAS session to cause SAS to read the entire existing tape library (for example, PROC CONTENTS). This step appends subsequent output to that tape library to the end of the physical sequential file. If you omit this step, output to that tape library overwrites the existing physical sequential file.

In one special case, CMS TAPE commands are needed with LABEL=NL. To initialize a new tape for LABEL=NL processing, you must issue

```
cms tape rew;
cms tape wtm 2;
```

Otherwise, the tape could run off the end.

Some tape processing examples using LABEL=NL include

1 copying a SAS data library to a new tape:

```
    /* insure tape position   */
cms tape rew;
    /* initialize tape        */
cms tape wtm 2;
    /* specify file 1         */
libname tapeout 'tap1' label=nl 1;
    /* copy SAS data library  */
proc copy in=lib1 out=tapeout;
    /*  from lib1 to tape     */
run;
```

2 copying multiple SAS data libraries to a single SAS data library on a new tape:

```
    /* insure tape position   */
cms tape rew;
    /* initialize tape        */
cms tape wtm 2;
    /* specify file 1         */
libname tapeout 'tap1' label=nl 1;
    /* copy lib1 to tape      */
proc copy in=lib1 out=tapeout;
    /* copy lib2 to the       */
proc copy in=lib2 out=tapeout;
    /*  same tape library     */
run;
```

3 copying a SAS data library to tape as a second sequential file (following a preexisting file):

```
    /* specify file 2        */
libname tapeout 'tap1' label=nl 2;
    /* copy SAS data library  */
```

```
proc copy in=lib2 out=tapeout;
    /*  from lib2 to a        */
run;
    /*  separate tape library */
```

4 appending a SAS data library to an existing SAS data library on a tape:

```
    /* specify file 1         */
libname tapeout 'tap1' label=nl 1;
    /* read the library       */
proc contents data=tapeout._all_ nods;
run;
    /* append SAS data library*/
proc copy in=lib3 out=tapeout;
    /*  from lib3 to tape     */
run;
```

Note: PROC CONTENTS is used to force SAS to load the tape's library structure so that PROC COPY will append rather than overwrite. △

LABEL=SL Processing Like LABEL=NL, LABEL=SL frees you from the need to position the tape manually. Standard label processing has advantages that are beyond the scope of this discussion.

With the LABEL=SL *n* option of the LIBNAME statement, you specify the sequential number of the file on the tape. For example, LABEL=SL 3 causes the third sequential file to be opened. You do not need to use the CMS TAPE command to position the tape; when SAS opens the sequential file, CMS automatically rewinds the tape and positions it to the specified file. Because the tape is rewound during positioning, it will be positioned correctly regardless of the prior tape position.

If you do output to a tape library and then want to append to the same tape library, continue to use the existing libref. Do not reissue the LIBNAME statement. Likewise, once you have read from or written to a tape library, if you then want to reread it, no special steps are required.

If you want to use PROC COPY to append to a tape library that was written in a previous SAS session, or for which the libref has been deassigned, you must cause SAS to read the entire existing tape library (for example, PROC CONTENTS). This step appends subsequent output to that tape library to the end of the physical sequential file. If you omit this step, output to that tape library overwrites the existing physical sequential file.

In one special case, you may need CMS TAPE commands with LABEL=SL. To initialize a new tape for LABEL=SL processing, you must submit the following statements:

```
cms tape rew;
cms tape wvol1 volid;
```

However if your tape is cataloged in a tape management system, this step has probably been done for you automatically.

Some tape processing examples using LABEL=SL include

1 copying a SAS data library to a new tape:

```
    /* specify file 1         */
libname tapeout 'tap1' label=sl 1
    /* and identify VOLID     */
        volid=V00001;
    /* copy SAS data library  */
```

```
proc copy in=lib1 out=tapeout;
    /*  from lib1 to tape     */
run;
```

2 copying multiple SAS data libraries to a single SAS data library on a new tape:

```
  /* specify file 1          */
libname tapeout 'tap1' label=sl 1
  /* and identify VOLID     */
   volid=V00001;
  /* copy lib1 to tape      */
proc copy in=lib1 out=tapeout;
  /* copy lib2 to the       */
proc copy in=lib2 out=tapeout;
  /*  same tape library     */
run;
```

3 copying a SAS data library to tape as a second sequential file (following a preexisting file):

```
    /* specify file 2          */
libname tapeout 'tap1' label=sl 2
    /* and identify VOLID     */
        volid=V00001;
    /* copy SAS data library  */
proc copy in=lib2 out=tapeout;
    /*  from lib2 to tape     */
run;
```

4 appending a SAS data library to an existing SAS data library on a tape:

```
    /* specify file 1      */
libname tapeout 'tap1' label=sl 1
    /*  and identify VOLID    */
        volid=V00001;
proc contents data=tapeout._all_ nods;
    /* read the library       */
run;
    /* append SAS data library*/
proc copy in=lib3 out=tapeout;
   /*  from lib3 to tape     */
run;
```

Note: PROC CONTENTS is used to force SAS to load the tape's library structure so that PROC COPY will append rather than overwrite. △

See Also

- □ "Working with SAS Files on Tape" on page 36
- □ *SAS Procedures Guide*

CPORT

Writes SAS data sets and catalogs into a special format in a transport file

CMS specifics; specification of transport file

Syntax

PROC CPORT <*options*>;

Details

The CPORT procedure writes SAS data sets and catalogs into a transport format. Coupled with the CIMPORT procedure, PROC CPORT enables you to move catalogs and data sets from one operating environment to another.

The transport format is written to the location specified by the FILE= option. The value of the FILE= option can be a fileref defined in a FILENAME statement or a physical file specification enclosed in single quotation marks. If you do not specify the FILE= option, the reserved fileref SASCAT is used, if you have assigned it, or else the file SASCAT DATA is written to your first R/W accessed filemode.

The following example creates a transport file SASCAT DATA:

```
filename sascat clear;
libname myfile 'a';
proc cport library=myfile;
run;
```

Any of the following examples create the transport file PORT FILE M:

```
filename sascat 'port file m';
libname myfile 'a';
proc cport library=myfile;
run;


filename portout 'port file m';
libname myfile 'a';
proc cport librarty=myfile file=portout;
run;


libname myfile 'a';
proc cport library=myfile file='port file m';
run;
```

To write the transport file to tape, assign a fileref to a tape. It is recommended that you use specify BLKSIZE=8000, as follows:

```
filename sascat tape 'tap1' blksize=8000;
libname myfile 'a';
proc cport library=myfile;
run;
```

If you specify a physical tape name on the FILE= option then you must also specify the TAPE option. This method is not recommended because the default blocksize of 80 is used, as follows:

```
libname myfile 'a';
proc cport library=myfile file='tap1' tape;
run;
```

See Also

- □ *Moving and Accessing SAS Files across Operating Environments*
- □ "FILENAME" on page 227
- □ *SAS Procedures Guide*

DATASETS

Lists, copies, renames, and deletes SAS files, manages indexes for and appends SAS data sets in a SAS data library

CMS specifics: output generated by CONTENTS statement, data library information

Syntax

PROC DATASETS <*options*>;

Details

The DATASETS procedure is part of base SAS software, but two portions of its output are system-dependent. The SAS data library information displayed in the SAS log depends on the operating environment. The CONTENTS statement in the DATASETS procedure generates the same output, including Engine/Host Dependent Information, as the CONTENTS procedure.

See Also

- □ "CONTENTS" on page 189
- □ *SAS Language and Procedures: Usage*
- □ *SAS Procedures Guide*

DBF

Converts a dBASE file to a SAS data set or a SAS data set to a dBASE file

CMS specifics: all

Syntax

PROC DBF *options*;

options

DB2 | DB3 | DB4 | DB5=*fileref*
  is the logical name to be associated with the external file or device type specified. The fileref can be a maximum of eight characters. The first character must be a

letter (A through Z), or an underscore (_). The remaining characters can be any of these characters or numerals (0 through 9).

The DB*n* option must correspond to the version of dBASE with which the DBF file is compatible. Specify a DBF file with the DB*n* option, where *n* is 2, 3, 4, or 5. You can only specify one of these values.

DATA=<*libref.*>*member*

names the input SAS data set, using 1–32 characters. Use this option if you are creating a DBF file from a SAS data set. If you use the DATA= option, do not use the OUT= option. If you omit the DATA= option, SAS creates an output SAS data set from the DBF file.

OUT=<*libref.*>*member*

names the SAS data set that is created to hold the converted data, using 1–32 characters. Use this option only if you do not specify the DATA= option. If OUT= is omitted, SAS creates a temporary data set in the WORK library. The name of the temporary data set is DATA1 [...DATA*n*]. If OUT= is omitted or if you do not specify a two-level name in the OUT= option, the SAS data set that is created by PROC DBF remains available during your current SAS session (under the temporary data set name), but it is not permanently saved.

Details

You can use PROC DBF in the CMS environment if your site has a license for SAS/ ACCESS for PC File Formats. To see a list of your licenses, submit the following statements:

```
proc setinit; run;
```

If you are licensed you will see an entry in your SAS log for SAS/ACCESS for PC File Formats.

The DBF procedure converts files in DBF format to SAS data sets that are compatible with the current SAS release. You can also use PROC DBF to convert SAS data sets to files in DBF format.

Before you convert a DBF file to a SAS file, you must first upload your DBF file from the Windows, OS/2, NT, or UNIX environment to the CMS environment, using a mechanism such as FTP (file transfer protocol). If you are licensed for SAS/CONNECT, you can use PROC UPLOAD:

```
filename out1 'sasdemo.emp.dbf';
proc upload infile='c:\employee\emp.dbf'
   outfile=out1 binary;
run;
```

PROC DBF produces one output file but no printed output. The output file contains the same information as the input file but in a different format.

The DBF procedure works with DBF files created by all the current versions and releases of dBASE (II, III, III PLUS, IV, and 5.0) and with most DBF files that are created by other software products.

Converting DBF Fields to SAS Variables

When you convert a DBF file a to SAS data set, DBF numeric variables become SAS numeric variables. Similarly, DBF character variables become SAS character variables. Any DBF character variable of length greater than 200 is truncated to 200 in SAS. Logical fields become SAS character variables with a length of 1. Date fields become SAS date variables.

DBF fields whose data are stored in auxiliary files (Memo, General, binary, and OLE data types) are ignored in SAS.

If a DBF file has missing numeric or date fields, SAS fills those missing fields with a series of the digit '9' or with blanks, respectively.

When a dBASE II file is translated into a SAS data set, any colons in dBASE variable names are changed to underscores in SAS variable names. Conversely, when a SAS data set is translated into a dBASE file, any underscores in SAS variable names are changed to colons in dBASE field names.

Example 1: Converting a dBASE IV File to a SAS Data Set

In this example, a dBASE IV file named SASDEMO.EMPLOYEE is converted to a SAS data set. A FILENAME statement specifies a fileref that names the dBASE IV file. The FILENAME statement must appear before the PROC DBF statement.

```
libname save 'sasdemo.employee.data';
filename dbfin 'sasdemo.employee dbf a';
proc dbf db4=dbfin out=save.employee;
run;
```

Example 2: Converting a dBASE 5 file to a SAS Data Set

In this example, a SAS data set is converted to a dBASE 5 file.

```
libname demo 'sasdemo.employee.data';
filename dbfout 'newemp dbf a' recfm=n;
proc dbf db5=dbfout data=demo.employee;
run;
```

Converting SAS Variables to DBF Fields

In DBF files, numeric variables are stored in character form. When converting from a SAS data set to a DBF file, SAS numeric variables become DBF numeric variables with a total length of 16. A SAS numeric variable with a decimal value must be stored in a decimal format in order to be converted to a DBF numeric field with a decimal value. In other words, unless you associate the SAS numeric variable with an appropriate format in a SAS FORMAT statement, the corresponding DBF field will not have any value to the right of the decimal point. You can associate a format with the variable in a SAS data set when you create the data set or by using the DATASETS procedure (see "DATASETS" on page 201).

If the number of digits—including a possible decimal point—exceeds 16, a warning message is issued and the DBF numeric field is filled with a series of the digit '9'. All SAS character variables become DBF fields of the same length. When converting from a SAS data set to a DBF file that is compatible with dBASE III or later, SAS date variables become DBF date fields. When converting to a dBASE II file, SAS date variables become dBASE II character fields in the form YYYYMMDD.

Transferring Other Software Files to DBF Files

You might find it helpful to save another software vendor's file to a DBF file and then convert that file into a SAS data set. For example, you could save an Excel XLS file in DBF format, upload the file, and use PROC DBF to convert that file into a SAS data set. Or you could do the reverse; use PROC DBF to convert a SAS data set into a DBF file and then load that file into an Excel spreadsheet.

ITEMS

Builds a SAS itemstore, which is essentially a file system within a file

CMS specifics: all

Syntax

PROC ITEMS NAME=<*libref.*>*member*;

NAME
If no libref is specified, the libref is assumed to be WORK. If *libref.member* is specified, the libref must have been previously allocated. See "LIBNAME" on page 243 for details.

Details

An *itemstore* is a SAS data set that is made up of independently accessible chunks of information. SAS uses itemstores for online help, where the SAS help browser accesses an itemstore in the SASHELP library. You can use the ITEMS procedure to create, modify, and browse your own itemstores, which you can access through the SAS help browser.

The contents of an itemstore are divided into directories, subdirectories, and topics. The directory tree structure emulates that of UNIX System Services, so that a given help topic is identified by a directory path (root_dir/sub_dir/item). This hierarchical structure allows the SAS help browser to supports HTML links between help topics.

The itemstores that SAS uses for HTML help can be written only by users with appropriate privilege. Though SAS Institute discourages rewrites of SAS help items, you can add items to the SAS help itemstores, and you can develop new itemstores of your own for any information that you wish to make available through the SAS help browser. For further information on writing your own HTML help, see "Developing User-Defined Help" on page 107.

To access an itemstore, you must first allocate the library that contains the itemstore, unless the itemstore is a member of the WORK library. After you allocate the library, you issue the PROC ITEMS NAME=*fileref* statement to access the itemstore in SAS. Once the itemstore is available in SAS, you can use the LIST, IMPORT, EXPORT, MERGE, and DELETE statements to control itemstore contents. SAS applies all of these statements to the itemstore name in the last PROC ITEMS NAME= statement.

For information on the HTML tags that are supported by the SAS help browser, see "Developing User-Defined Help" on page 107 .

HTC File Format

An HTC file is a collection of HTML files. HTC files can be imported and exported in HTC file format, where items and directories are separated by lines beginning with five colons:

```
:::::<filename>.htm
```

Directories in the HTC file are identified by a line that begins with five colons and ends with a path specification:

```
:::::<dirname1>/<dirname2>/<filename>.htm
```

In the previous example, if the HTC file containing this entry were imported, the directory and subdirectory would be created as needed and the file would be placed in the specified subdirectory. Any filename that lacks a path specification will go into the root directory or into the directory specified by the DIR= option, if it is specified.

Alternate Syntax for the DIR= and ITEM= Statements

You can use the forward slash path character (/) to specify a path in the DIR= and ITEM= options (described below) in all of the statements that take those options. For example, the following two statements are equivalent:

```
LIST DIR='usr' ITEM='mail';
LIST ITEM='usr/mail';
```

Note that a full path, starting with the directory just beneath the itemstore's root directory (with no initial forward slash) is required for access to anything except items in the root directory or to itemstores consisting of a single item.

Wildcards, using asterisks (*) as in UNIX, are not accepted in itemstore paths. Nor can you specify more than one path (a file concatenation) for each of the following statements.

LIST Statement

LIST<*options*;>

The LIST statement writes a list of item or directory names to the SAS log or to a specified file. Specifying no options writes a list of all items and directories to the SAS log.

Options

DUMP=*fileref*
specifies the fileref that will receive the listing. If DUMP= is not specified, the output goes to the SAS log.

DIR='*dir-name*'
specifies an itemstore directory whose item names you wish to list. If you specify the DIR= option alone, you will receive a listing of item names contained in that directory.

ITEM='*item-name*'
specifies that you wish to list the contents of the named item in the named directory of the itemstore. If you specify an item without specifying a directory, you will receive the contents of the item with the specified name in the root directory of the itemstore.

IMPORT Statement

The IMPORT statement imports a fileref into an itemstore. If the imported fileref contains items or directories that currently exist in the itemstore, the new items or directories overwrite (replace) the existing versions.

IMPORT FILEREF=*fileref*<*options*>;

Options

DIR='*dir-name*'
specifies the itemstore directory that will receive the imported fileref. If a directory is not specified, the fileref is imported into the root directory of the itemstore.

ITEM='*item-name*'
: specifies the name of the item that will receive the imported fileref. If an item is not specified, the imported fileref is assumed to be an HTC file.

EXPORT Statement

The EXPORT statement copies an item or itemstore to an external fileref in HTC format. If the fileref exists prior to the EXPORT statement, the new fileref overwrites (replaces) the previous version.

EXPORT FILEREF=*fileref*<*options*>;

Options

DIR='*dir-name*'
: specifies the itemstore directory that is the source of the export. If you do not specify a directory, the fileref receives the contents of the entire itemstore or the specified item from the root directory of the itemstore.

ITEM='*item-name*'
: specifies an item for export. If you do not specify an item, the fileref receives the entire contents of the specified directory or itemstore, in HTC format.

MERGE Statement

The MERGE statement merges the specified itemstore into the itemstore opened previously with PROC ITEMS.

MERGE SOURCE=<*libref.*>*member*;

A libref is required in the MERGE statement. If the two itemstores have directories with the same name and path, the contents of the new directory replace the contents of the old directory. If you merge into the root directory, the entire itemstore is replaced. If you merge a new item into a directory, the new item is merged into the old directory. If the old directory contains an item of the same name, the new item replaces the old item.

DELETE Statement

The DELETE statement deletes all or part of the contents in an itemstore.

DELETE<*options*>;

Options

DIR='*dir-name*'
: specifies the directory from which you wish to delete. When DIR= is not specified, either the entire contents of the itemstore are deleted or the specified item is deleted from the itemstore's root directory.

ITEM='*item-name*'
: deletes the specified item from the specified directory in the itemstore, or, if a directory is not specified, from the root directory of the itemstore.

See Also

- □ Information on the HELPLOC= system option in *SAS Language Reference: Dictionary*.
- □ “Developing User-Defined Help” on page 107 .

OPTIONS

Lists the current values of all SAS system options

CMS specifics: host options displayed

Syntax

PROC OPTIONS <*options*>;

Details

The session and configuration options displayed by the OPTIONS procedure are the same for every operating environment. However, the host options that are displayed with the HOST option are operating-environment dependent.

See Also

- □ “Summary Table of SAS System Options” on page 308
- □ “OPTIONS Procedure” on page 18
- □ *SAS Procedures Guide*
- □ *SAS Language Reference: Dictionary*
- □ *SAS Language and Procedures: Usage*

PMENU

Defines PMENU facilities for windows created with SAS software

CMS specifics: Some portable statements are ignored

Syntax

PROC PMENU <*options*>;

Details

The following statements and options are accepted without generating errors, but they have no effect under CMS:

- ACCELERATE= option in the ITEM statement
- MNEMONIC= option in the ITEM statement
- GRAY option in the ITEM statement
- HELP= option in the DIALOG statement.

See Also

- *SAS Procedures Guide*

PRINTTO

Defines destinations for SAS procedure output and the SAS log

CMS specifics: UNIT= option

Syntax

PROC PRINTTO <UNIT=*nn*> <LOG=*destination*> <PRINT=*destination*>;

Details

The UNIT= option sends printer listing output to the file identified by the fileref FT*nn*F001. If no fileref is defined, the print file is written to FT*nn*F001 LISTING.

See Also

- "Routing to External Files with the PRINTTO Procedure" on page 87
- *SAS Language and Procedures: Usage*
- *SAS Procedures Guide*

SORT

Sorts observations in a SAS data set

CMS specifics: SORT procedure statement options

Syntax

PROC SORT <*options*>;

Details

Under CMS, the SORT procedure uses the EBCDIC collating sequence. For general information about the SORT procedure see *SAS Procedures Guide*.

You can direct the SORT procedure to use either the SAS sort program, available under CMS and under all other operating environments, or a sort utility specific to CMS, or you can allow SAS to choose the best sort program to use. You make this choice with the SORTPGM= system option. See "System Options in the CMS Environment" on page 252 for details on the SORTPGM= option and the other system options that affect the SORT procedure. These system options include the following:

FILSZ
: specifies whether the SORT procedure being used supports the FILESIZE parameter. See "FILSZ" on page 268.

SORTCUT=
: specifies the number of observations above which the external sort program is selected when the SORTPGM=BEST system option is specified. See "SORTCUT=" on page 292.

SORTCUTP=
: specifies the data set size (in bytes) above which the sort program is selected when the SORTPGM=BEST option is specified. See "SORTCUTP=" on page 293.

SORTEQOP
: specifies whether the SORT procedure being used supports the EQUALS parameter. See "SORTEQOP" on page 294.

SORTLIB=
: specifies the TXTLIB to be made global for PROC SORT use. See "SORTLIB=" on page 294.

SORTLIST
: indicates whether SAS is to pass the LIST parameter to the SORT package being used (to request additional information about the sort process). See "SORTLIST" on page 295.

SORTMSG
: indicates whether SAS is to indicate to the SORT package to write all messages or just those that are critical. See "SORTMSG" on page 296.

SORTNAME=
: specifies the name of the host sort utility. See "SORTNAME=" on page 296.

SORTPARM=
: specifies a string to be appended to the OPTION statement that is passed to the host sort program. See "SORTPARM=" on page 297.

SORTPGM=
: specifies the name of the host sort program. See "SORTPGM=" on page 297.

SORTSIZE=
: specifies what value SAS is to pass to the SORT package being used to indicate the maximum virtual storage to be used. See "SORTSIZE=" on page 298

SORTSUMF
: indicates whether the SORT package being used supports the SUM FIELDS= parameter. See "SORTSUMF" on page 299.

SORT31PL
: controls whether SAS software calls the host sort program by using the extended (31-bit) plist or standard (24-bit) plist. See "SORT31PL" on page 299.

SORT Procedure Statement Options

The following host-specific sort options are available in the PROC SORT statement under CMS in addition to the sort options available on all hosts. Note that the list includes the portable EQUALS option because it has aspects specific to CMS.

DIAG
: passes the DIAG option to the host sort program. The interpretation of the DIAG option is entirely up to the host sort program; it has no effect on SAS. This option is recognized only when the system option SORT31PL is in effect.

EQUALS | NOEQUALS
: passes the EQUALS or NOEQUALS option to the host sort program regardless of the setting of the SAS system option SORTEQOP.

LEAVE=n
: adjusts the value of the storage parameter passed to the host sort program.

If SORTSIZE=SIZE is in effect, SAS estimates the size of the largest contiguous block of available storage, then subtracts the values of the system option LEAVE= and of the SORT procedure statement option LEAVE=, and passes the resulting value to the host sort. Note that the calculations are based on the size of the largest block of storage, whether it is above or below the 16M line. If the host sort is AMODE 24, however, SAS will consider only storage below the line. Some host sorts may allocate work storage only below the line even if they are AMODE 31, in which case the size estimated by SAS can be misleading.

If SORTSIZE=*n* is in effect, the LEAVE= value is subtracted from the SORTSIZE value.

For other values of SORTSIZE=, the LEAVE= option has no effect.

LIST | L
: passes the LIST option to the host sort program. The interpretation of the LIST option is entirely up to the host sort program; it has no effect on SAS. This option is ignored unless the system option SORT31PL is in effect.

MESSAGE | M
: passes the PRINT=ALL option to the host sort program. The MESSAGE option is useful if you run PROC SORT and the SAS log prints a message that the sort did not work properly. Explanations of the message can be found in the IBM or vendor reference manual that describes your system sort utility.

SORTSIZE=*n* | *n*K | *n*M | *n*G | MAX | SIZE | *n*K | *n*M | *n*G | MAX | SIZE
: specifies the maximum virtual storage that can be used by the system sort utility. If not specified, the default is given by the SORTSIZE= SAS system option.

See Also

- "System Options in the CMS Environment" on page 252
- *SAS Language and Procedures: Usage*
- *SAS Procedures Guide*

TAPECOPY

Copies an entire tape volume or files from one or more tape volumes to a single output tape volume

CMS specifics: all

Syntax

PROC TAPECOPY <*options*>;
  <**INVOL** <*options*> </'*CMSmount*'>;>
  <**FILES***file-numbers*>;

Details

The TAPECOPY procedure copies an entire tape volume or files from one or more tape volumes to a single output tape volume. Use the TAPECOPY procedure to copy standard labeled or nonlabeled 9-track tapes and 18-track tape cartridges under CMS. PROC TAPECOPY always begins writing at the beginning of the output tape volume.

CAUTION:
Any files that exist on the output tape prior to the copy operation are destroyed. △

You can specify, within limits, whether the output tape will be standard labeled (SL) or nonlabeled (NL). You cannot create an SL tape using an NL input tape because PROC TAPECOPY cannot manufacture tape labels. However, you can create a nonlabeled output tape volume from a labeled input tape. Under CMS, PROC TAPECOPY writes over any existing labels on the output tape.

PROC TAPECOPY Statement

PROC TAPECOPY <*options*>;

The PROC TAPECOPY statement accepts the following options.

COPYVOLSER
specifies that the output tape should have a standard label with the same volume serial as the first input tape. This option is effective only when the output tape volume is to be standard labeled, that is, LABEL=SL. This option is the default. Use the NEWVOLSER option to specify a different volume serial.

DEN=*density*
specifies the density of the output tape. (The DEN= option should not be specified for cartridge tapes.) If the DEN= option appears in the PROC TAPECOPY statement, it overrides any density specification in the FILEDEF for the output tape volume. If you do not specify a density in the PROC TAPECOPY statement, the tape is written at the highest density possible for that tape.
Valid density values are given by the following table.

Tape Density Value	Tape Volume Type
DEN=2	800 bpi
DEN=800	
DEN=3	1600 bpi
DEN=1600	

Tape Density Value	Tape Volume Type

DEN=4
DEN=6250
	6250 bpi

DEN=5
DEN=38K
	38K bpi

DETACH
: requests that all tape drives that are used by PROC TAPECOPY be detached after the procedure has executed.

INDD=*fileref*
: specifies the fileref of the first input tape volume. The default INDD= value is VOLIN.

INVOL=*volume-serial*
: specifies the volume serial of the first input tape. This option is valid only when the input tape is standard labeled.

LABEL=SL | NL | BLP
: specifies whether the output tape volume is to be standard labeled (LABEL=SL) or nonlabeled (LABEL=NL), or that all label processing should be bypassed (LABEL=BLP). The default LABEL= option value is BLP.

Note: Do not specify SL if you intend to copy any nonlabeled tapes. △

If SL is specified, the output tape is SL; if NL is specified, the output tape is NL. When BLP is specified, all files are treated as physical files and the distinction between data files and label files is irrelevant; therefore, the output tape has the label status of the input tape.

In the following example an unlabeled output tape is created from a standard labeled input tape using PROC TAPECOPY:

```
proc tapecopy label=nl nolist;
run;
```

After PROC TAPECOPY has executed, the output tape volume is a nonlabeled tape. In this example, specifying LABEL=NL was necessary. If it were not specified, the default of LABEL=BLP would have been used, and all files, including data and labels, would have been copied.

NEWVOLSER=*new-volume-serial*
: specifies a new volume serial for the output tape. NEWVOLSER is effective only if the output tape is to be standard labeled.

NOFSNRESEQ
: specifies that file sequence numbers in the file labels should not be resequenced when a standard labeled output tape volume is being produced. PROC TAPECOPY normally resequences these numbers and updates the label to reflect the ordinal position of the file on the output tape as it is copied and the actual density at which the output tape is written. NFR is an alias for the NOFSNRESEQ option.

NOLIST
: suppresses printing of the tape characteristics and the summary of copied files in the log. Regardless of whether you specify NOLIST, the SAS log contains a brief summary of the action produced by PROC TAPECOPY; this summary is usually

sufficient to verify proper functioning of the TAPECOPY procedure if you are familiar with the contents of the input tape(s).

OUTDD=*fileref*
specifies the fileref of the output tape. The default OUTDD= value is VOLOUT.

TAP1
specifies tape drive 181 as the input tape device. There is no need for an input tape FILEDEF if this option is used, as TAP1 is the default. If INDD= is not specified and there is no FILEDEF for VOLIN, TAP1 is the default.

Note: Do not specify both the INDD= and TAP1 options. △

TAP*n*
specifies a CMS tape symbolic name as the output device, where *n* can be a value of 0, 2–9, or A–F. See Table 18.1 on page 230 for a table listing the symbolic names and the corresponding virtual addresses for CMS tape output devices.

A corresponding FILENAME statement or CMS FILEDEF command is unnecessary if the TAP*n* option is used. If the OUTDD= option has not been specified and there is no FILEDEF for VOLOUT, the TAP*n* value defaults to TAP2.

Note: Do not specify both the OUTDD= and TAP*n* options. △

Site-specific Limitations Some CMS sites do not have CMS tape mount commands but do allow you to mount tapes by contacting the computer operator. If your installation does not support CMS tape mount commands, you can still use PROC TAPECOPY, but you cannot specify deferred tape mounting for input tapes. If your installation has CMS tape mount commands, you can specify deferred tape mounting with an option in the INVOL statement.

Note: PROC TAPECOPY supports deferred mounting of input tapes only. △

Using Default Values If you do not specify any options in the PROC TAPECOPY statement (and there are no FILEDEF commands for the VOLIN and VOLOUT options), PROC TAPECOPY defaults to TAP1 for input and TAP2 for output. The output tape volume is labeled with the same label as the input tape. All files, including labels, are copied because LABEL=BLP is assumed.

Using the FILENAME Statement with the TAPECOPY Procedure You can use the FILENAME statement to associate filerefs with your input and output tapes. For example, the following code copies a tape to a 1600 bpi tape. LABEL=BLP is assumed as the default for the label status of both tapes.

```
filename intape tape 'tap1';
filename outtape tape 'tap2';

proc tapecopy indd=intape outdd=outtape den=1600 nolist;
run;
```

Printed Output

The TAPECOPY procedure prints a listing to the SAS log of the input and output tape characteristics and a summary of the files copied.

INVOL Statement

INVOL *<options> <'CMS-mount-command'>;*

The INVOL statement defines an input tape volume from which some or all files are to be copied to the output tape volume. The INVOL statement is not necessary if you are using only one input tape or if it is the first of several input tapes. For either of the previous cases, you should use the INDD= and INVOL= options in the PROC TAPECOPY statement instead. However, when you are using several input tapes, use an INVOL statement for each tape after the first input tape.

If you want to use deferred mounting for an input tape, you must use an INVOL statement with the /*'CMS-mount-command'* option.

Note: You cannot use deferred mounting for an output tape. △

The following options can appear in the INVOL statement:

BLP
: specifies that label processing is to be bypassed. BLP is the default if neither SL nor NL is specified.

Note: Be sure that you know the contents of any tape for which you specify BLP in an INVOL statement to avoid copying labeled and nonlabeled tapes to the same output tape. △

/*'CMS-mount-command'*
: specifies the site-specific tape mount command that your installation uses to mount a tape. Follow the slash with the text of the mount command enclosed in single quotes. This option causes deferred mounting of the input tape; it must be used if you want deferred mounting. If this option is used, it must be specified as the last option in the INVOL statement.

The mount request is executed via the standard CMS function call (SVC 204). If the return code is not zero (for example, if PROC TAPECOPY cannot find the specified mount command), SAS prints a message on the SAS log and the procedure stops processing. Some mount commands require that the tape drive be detached before the mount is issued. If this is a requirement, then you must specify DETACH in the INVOL statement.

Note: At installations that do not have any mount commands, this option is invalid; therefore, you cannot use deferred mounting. △

DETACH
: specifies that the tape drive be detached before issuing a mount command.

INDD=*fileref*
: specifies the fileref of the current input tape. The default INDD= value is the fileref already in effect for the previous input tape volume, as specified in the PROC TAPECOPY statement or the last INVOL statement.

INVOL=*volume-serial*
: specifies the volume serial of the current input tape. This option is valid only when the input tape is standard labeled.

NL
: defines the input tape as nonlabeled. If the input tape is actually standard labeled, specifying the NL option causes the tape to be treated as if it were nonlabeled. In this case, any file numbers specified in FILES statements must be physical file numbers, not logical file numbers.

If you specify LABEL=SL in the PROC TAPECOPY statement (for the output tape), do not specify NL on a subsequent INVOL statement. In other words, do not copy labeled and nonlabeled tapes onto the same output tape unless the labeled tapes are to be treated as nonlabeled.

SL
: specifies that the input tape is standard labeled. Do not specify SL unless the tape is actually standard labeled.

TAP*n*

specifies the tape drive to use, where *n* can be a value of 0, 2–9 or A–F.

The default value is the TAP*n* in effect from the PROC TAPECOPY statement or previous INVOL statement.

Do not use both the INDD= and TAP*n* options.

Copying Multiple Files from Multiple Tapes

In the following example, files are copied from four standard labeled input tapes to one output tape (which becomes standard labeled):

```
proc tapecopy tap1 nolist tap2 copyvolser label=sl;
   invol sl /'mount t13794 on 181 noring';
   file 3;
   invol sl /'mount txxxxx on 181 noring';
   files 4 5-7;
   invol sl /'mount tyyyyy on 181 noring';
   files 2 4 1 9-eov;
run;
```

An INVOL statement is used for each input tape to specify that the tapes are standard labeled and to provide the deferred tape mount command. Each tape must be standard labeled, or PROC TAPECOPY fails.

FILES Statement

FILES *file-numbers*;

Use the FILES statement when you want to copy individual files from an input tape. The FILES statement allows you to specify the files you want to copy. You can use as many FILES statements as you need to specify particular files. Depending on the kind of tape (labeled or nonlabeled) being copied, and the intended label status of the output tape, you specify either physical or logical file numbers in the FILES statement. The following table shows the correspondence between the type of input and output tape you are using and the kind of file numbers you should specify.

Table 17.1

Input Tape	LABEL= Value from PROC TAPECOPY Statement	Output Tape	File Numbers
SL	NL	NL	logical
SL	SL	SL	logical
NL	NL	NL	physical
NL	BLP	NL	physical
SL	BLP	SL	physical

Note: A physical file is defined as the information on a tape between two tape marks. A logical file actually consists of three physical files; the first contains a header label, the second contains the data, and the third contains a trailer label. The term logical file in this context implies a standard labeled tape. △

When you select specific files from the first input tape, the FILES statement(s) directly follows the PROC TAPECOPY statement. When you use several input tape volumes, follow each INVOL statement with its associated FILES statement(s).

Specifying Individual Files File numbers in a FILES statement can be specified in any order. If, for example, you want to copy file 5 and follow it by file 2 and then file 1, you can specify the following:

```
proc tapecopy;
   files 5 2;
   files 1;
run;
```

Or, you can specify the following:

```
proc tapecopy;
   files 5 2 1;
run;
```

Specifying a Range You can specify a range of files by putting a dash between the two files, as in the following example:

```
proc tapecopy;
   files 1-7;
run;
```

In a range, the second number must be greater than the first. The keyword EOV can be used as the second file in a range. PROC TAPECOPY copies all files on the input tape until the end of the volume is reached, which is, in most cases, indicated by a double tape mark.

TAPELABEL

Lists the label information of an IBM standard labeled tape volume

CMS specifics: all

Syntax

PROC TAPELABEL <*options*>;

Details

The TAPELABEL procedure lists the label information of an IBM standard labeled tape volume. The TAPELABEL procedure can process one or more standard labeled tape volumes. The procedure prints information from the tape label, including the data set name, DCB information, and data set history. Alternately, you can use the MAP option to print information about standard labeled and nonstandard labeled tapes, including block sizes, block counts, and length of tape in feet for each file.

PROC TAPELABEL Statement

PROC TAPELABEL <*options*>;

The following options can appear in the PROC TAPELABEL statement:

DDNAME=(*fileref-1... fileref-n*)
specifies the fileref assigned to the tape volume to be processed. More than one fileref can be specified. If you specify only one fileref, you can omit the parentheses.

DUMP
provides, in addition to normal output a dump of the first 80 bytes of the first 10 blocks of each file.

MAP
treats the type as nonlabeled and provides the following information for every file on the tape. This information replaces the output listed in "Printed Output" on page 217:

- □ the number of blocks
- □ the length in bytes of the largest and the smallest block in the file
- □ the density with which the tape was written
- □ the length in feet.

PAGE
begins the output for each tape volume on a new page.

TAP*n*
specifies a CMS tape symbolic name, where *n* can be a value of 0, 1–9, or A–F. See Table 18.1 on page 230 for a listing of the virtual address that corresponds to each symbolic name.

This option can be used instead of (or in addition to) the DDNAME= option to specify a tape volume to be processed. A corresponding FILENAME statement or CMS FILEDEF command is unnecessary if the TAP*n* option is used.

Note: If you omit the DDname= and TAP*n* options, the default fileref is TAP1. △

Printed Output For each file on a tape volume, the TAPELABEL procedure prints the following information:

- □ FILE NUMBER, the file sequence number (from header label)
- □ DSNAME, the data set name
- □ RECFM, the record format
- □ LRECL, the logical record length
- □ BLKSIZE, the block size
- □ BLOCK COUNT, the number of blocks in the file (from the trailer label)
- □ EST FEET, the estimated length of the file in feet
- □ CREATED, the file creation date
- □ EXPIRES, the file expiration date
- □ CREATED BY JOB NAME STEPNAME, the job and step names of the job that created the file
- □ TRTCH, the track recording technique
- □ DEN, the file recording density code
- □ PSWD, the file protection indicator
- □ UHL, the number of user header labels
- □ UTL, the number of user trailer labels.

In addition, the TAPELABEL procedure prints the sum of the estimated file lengths.

Examples

Use the TAPELABEL procedure to list label information. For example, the following statements list the label information for all files on the tape volume currently mounted on virtual address 182:

```
proc tapelabel tap2;
   title 'Label Information for VA182 Tape Volume Files';
run;
```

If you have tape volumes mounted at multiple virtual addresses, use statements similar to the following:

```
filename labtape tape 'tap3';

proc tapelabel ddname=labtape tap7 tapf page;
run;
```

These statements list the label information for all files on the tape volumes currently mounted on virtual addresses 183, 187, and 28F. The output for each volume is printed on a separate page.

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

SAS® Companion for the CMS Environment, Version 8

ISBN 1–58025–481–0

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

1st printing, October 1999