CHAPTER

16

Procedures

SAS Procedures under OpenVMS

Base SAS procedures allow you to perform statistical computations, create reports, and manage your data. Most of the base SAS procedures are completely described in *SAS Procedures Guide*. The procedures described here have syntax or behavior that is specific to the OpenVMS operating environment.

CATALOG

Manages entries in SAS catalogs

Language element: procedure

OpenVMS specifics: FILE= option in the CONTENTS statement

Syntax

PROC CATALOG CATALOG=<*libref.*>*catalog* <ENTRYTYPE=*etype*> <KILL>;
CONTENTS <OUT=*SAS-data-set*> <FILE=*fileref*>;

Note: This is a simplified version of the CATALOG procedure syntax. For the complete syntax and its explanation, see the CATALOG procedure in *SAS Procedures Guide*. △

fileref
names a file specification that is specific to the OpenVMS operating environment.

Details The CATALOG procedure manages entries in SAS catalogs.

The FILE= option in the CONTENTS statement of the CATALOG procedure accepts a file specification that is specific to OpenVMS. If an unquoted file specification is given in the FILE= option, but no FILENAME statement or DCL DEFINE command has been issued to define the file specification, then the file is named according to the rules for naming OpenVMS files. Consider the following example. If MYFILE is neither a SAS fileref nor an OpenVMS logical name, then the file MYFILE.LIS, containing the list of contents for SASUSER.PROFILE, is created in your default directory:

```
proc catalog catalog=sasuser.profile;
   contents file=myfile;
run;
```

See Also

□ CATALOG procedure in *SAS Procedures Guide*

CIMPORT

Restores a transport file created by the CPORT procedure

Language element: procedure
OpenVMS specifics: name and location of transport file; using a transport file on tape

Syntax

PROC CIMPORT *destination=libref.member-name<option(s)>*;

destination
specifies the name and location of a file to be transported.

Details The CIMPORT procedure *imports* a transfer file that was created (*exported*) by the CPORT procedure.

If you have used the CPORT procedure, the CIMPORT procedure allows you to move catalogs, data sets, and SAS data libraries from one operating environment to another.

Note: The CIMPORT procedure processes a transport file that was generated by PROC CPORT, not a transport file that was generated by the XPORT engine. △

Examples

Example 1: Importing an Entire Data Library from a Disk

```
libname newlib 'SAS-data-library';
filename tranfile 'transport-file';
```

```
proc cimport library=newlib infile=tranfile;
run;
```

PROC CIMPORT reads from disk the transport file TRANFILE that a previous PROC CPORT created from a SAS data library and restores the transport file to the SAS data library NEWLIB.

Example 2: Importing an Entire Data Library from a Tape

CAUTION:
You must use an unlabeled tape when reading from tape with the CIMPORT procedure. △

Mount the tape onto the tape drive.

```
x 'alloc tape-drive sastape';
x 'mount/for sastape';
libname newlib 'SAS-data-library';
filename tranfile 'sastape';
proc cimport library=newlib infile=tranfile tape;
run;
```

PROC CIMPORT reads from tape the transport file TRANFILE that PROC CPORT (using the TAPE option) created from a SAS data library and restores the transport file to the SAS data library NEWLIB.

See Also

- CIMPORT procedure in *SAS Procedures Guide*
- Procedure: “CPORT” on page 341
- *Moving and Accessing SAS Files across Operating Environments*

CONTENTS

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

Language element: procedure
OpenVMS specifics: Engine/Host Dependent Information output

Syntax

PROC CONTENTS <*option(s)*>;

Note: For a complete listing and explanation of the available options, see the CONTENTS procedure in *SAS Procedures Guide*. △

option(s)
can be the following:

DIRECTORY
prints a list of information specific to the OpenVMS operating environment. This information is the same as the PROC DATASETS directory information that is written to the log.

Details The CONTENTS procedure shows the contents of a SAS data set and prints the directory of the SAS data library.

Although most of the printed output that the CONTENTS procedure generates is the same on all operating environments, the **Engine/Host Dependent Information** output is specific to your operating environment. Output 16.1 on page 338 shows the **Engine/Host Dependent Information** generated for the V8 engine from the following statements:

```
proc contents data=oranges;
run;
```

Output 16.1 Engine/Host Dependent Information from PROC CONTENTS Using the V8 Engine

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
Filename:	SASDISK:[SASDEMO.SAS$WORK2040F93A]ORANGES.SAS7BDAT
Release Created:	8.00.00P
Host Created:	OpenVMS
File Size (blocks):	32

-----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

The engine name is listed in the header information. The **Engine/Host Dependent Information** describes the size of the data set, as well as the physical name of the data set.

See Also

- CONTENTS procedure in *SAS Procedures Guide*
- "Starting with SAS Data Sets" in *SAS Language and Procedures: Usage*

CONVERT

Converts OSIRIS and SPSS system files to SAS data sets

Language element: procedure
OpenVMS specifics: All aspects are host-specific

Syntax

PROC CONVERT *product-specification <option(s)>*;

product-specification
is required and can be one of the following:

OSIRIS=*fileref-1* DICT=*fileref-2*
specifies a fileref or libref for the OSIRIS file to be converted into a SAS data set. If you use this product specification, you must also use the DICT= option, which specifies the OSIRIS dictionary to use.

SPSS=*fileref*
specifies a fileref or libref for the SPSS file to be converted into a SAS data set. The SPSS file can have the following formats:

- SPSS-X format (whose originating operating environment is OpenVMS)
- Portable File Format (from any operating environment).

option(s)
can be one or more of the following:

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

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

OUT=*SAS-data-set*
names the SAS data set that is created to hold the converted data. If the OUT= option is omitted, the SAS System still creates a data set and automatically names it DATA*n*, as if you omitted a data set name in a DATA statement. If it is the first such data set created in a job or session, the SAS System names it DATA1; the

second is named DATA2, and so on. If the OUT= option is omitted, or if you do not specify a two-level name in the OUT= option, then the converted data set is stored in your WORK data library and by default is not permanent.

Details The CONVERT procedure converts an OSIRIS or SPSS data file to a SAS data set. It 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 in "Output Data Sets" on page 340. The OSIRIS and SPSS engines provide more extensive capabilities.

Because the OSIRIS and SPSS products are maintained by other companies or organizations, changes may be made that make the system files incompatible with the current version of PROC CONVERT. SAS Institute upgrades PROC CONVERT to support changes made to these products only when a new version of the SAS System is available.

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

Output Data Sets This section describes the attributes of the output SAS data set for each *product-specification* value.

CAUTION:

Be sure that the translated names are unique. Variable names can sometimes be translated by the SAS System. To ensure the procedure works correctly, be sure that your variables are named in such a way that translation results in unique names. △

OSIRIS Output For single-response variables, the V1–V9999 name becomes the SAS variable name. For multiple-response variables, the suffix R*n* is added to the variable name, where *n* is the response number. 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 that is longer than 200 is truncated to 200. The OSIRIS variable description becomes a SAS variable label, and OSIRIS print formats become SAS formats.

SPSS Output SPSS variable names and variable labels become unchanged variable names and labels. SPSS alphabetic variables become SAS character variables of length 4. SPSS blank values are converted into SAS missing values. SPSS print formats become SAS formats, and the SPSS default precision of no decimal places becomes part of the variables' formats. SPSS value labels are not copied. DOCUMENT data are copied so that PROC CONTENTS can display them.

Comparisons Using the CONVERT procedure is similar to using the OSIRIS and SPSS engines. For example, the following two programs provide identical results:

```
   /*  using the CONVERT procedure  */
filename xxx 'mybmdp.dat';
proc convert osiris=xxx out=temp;
run;


   /*  using the OSIRIS engine  */
libname xxx osiris 'myosiris.dat';
```

```
data temp;
    set xxx._first_;
run;
```

However, the OSIRIS and SPSS engines provide more extensive capability than PROC CONVERT.

Example

The following is an example of converting an OSIRIS file to a SAS data set, using a fileref named **save**:

```
filename save '[mydir]osiris.dat';

proc convert osiris=save;
run;
```

If you have more than one save file in the OSIRIS file referenced by *fileref*, then you can use two additional options in parentheses after the fileref. The CODE= option lets you specify the code of the save file that you want, and the CONTENT= option lets you specify the save file's content. For example, if a save file had CODE=JUDGES and CONTENT=DATA, you could use the following statements to convert the save file to a SAS data set:

```
filename save '[mydir]osiris1.dat';

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

See Also

□ "Using the OSIRIS and SPSS Engines" on page 161

CPORT

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

Language element: procedure
OpenVMS specifics: name and location of transport file; creating a transport file on tape

Syntax

PROC CPORT *source-type=libref<.member-name> <option-list>*;

Note: This is a simplified version of the CPORT procedure syntax. For the complete syntax and its explanation, see the CPORT procedure in *SAS Procedures Guide*. △

libref
specifies the name and location of the file to be transported.

Details If you do not use the FILE= option and have not defined the reserved fileref SASCAT, a file named SASCAT.DAT is created in your default directory.

The CPORT procedure defaults to writing to a file on disk. If you want to write to a file on tape, you must use the TAPE option in the PROC CPORT statement.

Note: You do not need to define the fileref SASCAT to your tape drive. You can choose any fileref, or you can let the file default to SASCAT.DAT, as it does for disk access. △

You do not need to use the /BLOCKSIZE=8000 option in the DCL MOUNT command, although it is recommended.

Examples

Example 1: Exporting Data Sets and Catalogs to Disk The following is an example of using PROC CPORT to export all the data sets and catalogs from a data library on disk:

```
libname newlib 'SAS-data-library';
filename tranfile 'transport-file';
proc cport library=newlib file=tranfile;
run;
```

PROC CPORT writes the file TRANFILE to disk. This file contains the data sets and catalogs in the SAS data library NEWLIB in transport format.

Example 2: Exporting Data Sets and Catalogs to Tape

CAUTION:
You must use an unlabeled tape when writing to tape with the CPORT procedure. △

The following is an example of using PROC CPORT to export all the data sets and catalogs in a data library on tape and then mount the tape onto the tape drive:

```
x 'alloc tape-drive sastape';
x 'mount/for sastape';
libname newlib 'SAS-data-library';
filename tranfile 'sastape';

proc cport library=newlib file=tranfile tape;
run;
```

PROC CPORT writes the file TRANFILE to tape. This file contains the data sets and catalogs in the SAS data library NEWLIB in transport format.

See Also

- CPORT procedure in *SAS Procedures Guide*
- Procedure: “CIMPORT” on page 336
- *Moving and Accessing SAS Files across Operating Environments*

DATASETS

Lists, copies, renames, repairs, and deletes SAS files; also manages indexes for and appends SAS data sets in a SAS data library; changes variable names and related variable information and prints the contents

Language element: procedure

OpenVMS specifics: directory information; CONTENTS statement output

Syntax

PROC DATASETS <*option(s)*>;
CONTENTS <*option(s)*>

Note: This is a simplified version of the DATASETS procedure syntax. For the complete syntax and its explanation, see the DATASETS procedure in *SAS Procedures Guide*. △

CONTENTS *option(s)*
the value for *option(s)* can be the following:

DIRECTORY
prints a list of information specific to the OpenVMS operating environment.

Details The SAS data library information that the DATASETS procedure displays in the SAS log is specific to your operating environment. Output 16.2 on page 343 shows the information that the DATASETS procedure writes to the SAS log when the following SAS statements are executed in the OpenVMS environment:

```
proc datasets library=work;
run;
```

Output 16.2 SAS Data Library Information from PROC DATASETS with the V8 Engine

```
1     proc datasets library=work;

                                      -----Directory-----

                            Libref:         WORK
                            Engine:         V8
                            Physical Name: SASDISK:[SASDEMO.SAS$WORK2542293E]

                                  #  Name      Memtype  Indexes
                                  ----------------------------
                                  1  ORANGES   DATA

2     quit;
$
```

The output shows you the libref, engine, and physical name associated with the library, as well as the names of the data sets that the library contains. It also shows the names of any catalogs and valid memtype stored in the library.

The CONTENTS statement of the DATASETS procedure generates the same **Engine/Host Dependent Information** as the CONTENTS procedure.

See Also

- DATASETS procedure in *SAS Procedures Guide*
- Procedure: "CONTENTS" on page 337
- "Modifying Data Set Names and Attributes" in *SAS Language and Procedures: Usage*

FORMAT

Creates user-defined formats and informats

Language element: procedure

OpenVMS specifics: location of temporary formats and informats

Syntax

PROC FORMAT <*options(s)*>;

Note: This is a simplified version of the FORMAT procedure syntax. For the complete syntax and its explanation, see the FORMAT procedure in *SAS Procedures Guide*. △

option(s)
specifies the options that control the behavior of and information about formats and informats.

Details The FORMAT procedure enables you to define your own informats and formats for variables.

Under OpenVMS, temporary formats and informats are stored in the temporary catalog FORMATS.SASEB$CATALOG in your WORK data library.

See Also

- FORMAT procedure in *SAS Procedures Guide*

OPTIONS

Lists the current values of all SAS system options

Language element: procedure

OpenVMS specifics: host options listed

Syntax

PROC OPTIONS <*options(s)*>;

option(s)
controls the format of the list of system options and the number of items displayed. The value for *option(s)* can be any of the following:

HOST | NOHOST
displays only host options (HOST) or only portable options (NOHOST). PORTABLE is an alias for NOHOST.

LONG | SHORT
specifies the format for displaying the settings of the SAS system options. LONG lists each system option on a separate line with an explanation. SHORT produces a compressed listing without explanations.

OPTION=*option-name* <DEFINE> <VALUE>
displays a short description and the value (if any) of the option specified by *option-name*. DEFINE and VALUE provide additional information about the option.

option-name
specifies the option to use as input to the procedure.

DEFINE
displays the short description of the option, as well as its type and how to get, set, and display its value.

VALUE
displays the option value and scope, as well as how the value was set.

Requirement: If a SAS system option uses an equals sign, such as PAGESIZE=, do not include the equals sign when specifying the option to the OPTION= procedure.

Details The OPTIONS procedure lists the current settings of the SAS system options.

The portable options (session and configuration) displayed by the OPTIONS procedure are the same for every operating environment, although the default values may differ slightly. However, the host options that PROC OPTIONS displays are different for each operating environment.

Also, some option values depend on which mode of operation you use to run SAS. For example, the default for the LOG= option is LOG under a windowing environment, but in interactive line mode the default is SYS$OUTPUT. Finally, the way you set up your process affects the default values of system options. For example, the default value of the CONFIG= option depends on whether you have defined the OpenVMS logical name SAS$CONFIG in your process.

By using PROC OPTIONS, you can check the values of all system options. For more information about a particular host option, see the entry for the option in Chapter 18, “System Options,” on page 387.

See Also

- OPTIONS procedure in *SAS Procedures Guide*

PMENU

Defines pull-down menu facilities for windows that were created with SAS software

Language element: procedure
OpenVMS specifics: Some options and statements are ignored in the OpenVMS environment

Syntax

PROC PMENU <CATALOG=<*libref.*>*catalog*>
<DESC *'entry-description'*>;

Note: This is a simplified version of the PMENU procedure syntax. For the complete syntax and its explanation, see the PMENU procedure in *SAS Procedures Guide*. △

CATALOG=<*libref.*>*catalog*
specifies the catalog in which you want to store PMENU entries. If you omit *libref*, the PMENU entries are stored in a catalog in the SASUSER data library. If you omit CATALOG=, the entries are stored in the SASUSER.PROFILE catalog.

DESC *'entry-description'*
provides a description of the PMENU catalog entries created in the step.

Details The PMENU procedure defines pull-down menus that can be used in DATA step windows, macro windows, SAS/AF and SAS/FSP windows, or in any SAS application that allows you to specify menus.

Under OpenVMS, the following statements or statement options are ignored:

- SEPARATOR statement
- HELP= option in the DIALOG statement
- ACCELERATE= and MNEMONIC= options in the ITEM statement
- ATTR= and COLOR= options in the TEXT statement. (The colors and attributes for text and input fields are controlled by the CPARMS colors. For details, see "Customizing Colors" on page 103.

The GRAY option makes any unavailable menu items look different (usually bold) from those that are available. On some displays, this visual distinction is not supported; on these displays, all menu items appear the same.

See Also

- PMENU procedure in *SAS Procedures Guide*

PRINTTO

Defines destinations for SAS procedure output and for the SAS log

Language element: procedure
OpenVMS specifics: valid values for *file-specification*; UNIT= option

Syntax

PROC PRINTTO <*option(s)*>

Note: This is a simplified version of the PRINTTO procedure syntax. For the complete syntax and its explanation, see the PRINTTO procedure in *SAS Procedures Guide*. △

option(s)

FILE=*file-specification*
specifies a fileref, a fully qualified OpenVMS pathname (in quotation marks), or an OpenVMS logical name. This is an alias for the PRINT= option.

LOG=*file-specification*
specifies a fileref, a fully qualified OpenVMS pathname (in quotation marks), or an OpenVMS logical name.

NAME=*file-specification*
specifies a fileref, a fully qualified OpenVMS pathname (in quotation marks), or an OpenVMS logical name. This is an alias for the PRINT= option.

PRINT=*file-specification*
specifies a fileref, a fully qualified OpenVMS pathname (in quotation marks), or an OpenVMS logical name.

UNIT=*nn*
sends output to the file FT*nn*F001.LIS, where *nn* represents the UNIT= value, which can range from 1 to 99.

Details The PRINTTO procedure defines destinations for SAS procedure output and for the SAS log.

To send output directly to a printer, use a FILENAME statement with the PRINTER device-type keyword. This sends the output to the default SYS$PRINT queue. If you want to override the default queue, use the QUEUE= option in the FILENAME statement to specify a different queue.

Note: You cannot send the output directly to a member of a text library or to a remote node or tape. △

Examples

Example 1: Sending SAS Log Entries to an External File The following statements send any SAS log entries that are generated after the RUN statement to the external file that is associated with the fileref MYFILE:

```
filename myfile '[mydir]mylog.log';

proc printto log=myfile;
run;
```

Example 2: Sending Procedure Output to an External File The following statements send the procedure output from the CONTENTS procedure (and any other subsequent

procedure output from the SAS session) to the external file that is associated with the OpenVMS logical name OUTPUT:

```
x 'define output [mydir]proc1.lis';
proc printto print=output;
run;

proc contents data=oranges;
run;
```

Example 3: Sending Procedure Output to a System Printer The following statements send the procedure output from the CONTENTS procedure directly to the system printer:

```
filename myfile printer '[mydir]proc2.lis';

proc printto print=myfile;
run;

proc contents data=oranges;
run;
```

Example 4: Redirecting SAS Log and Procedure Output to the Default The following statements (a PROC PRINTTO statement with no options) redirect the SAS log and procedure output to the original default destinations:

```
proc printto;
run;
```

Example 5: Sending Procedure Output to a File The following statements send any procedure output to a file named MYPRINT.DAT:

```
proc printto print=myprint;
run;
```

See Also

- PRINTTO procedure in *SAS Procedures Guide*
- Statement: "FILENAME" on page 357
- Chapter 8, "Routing the SAS Log and SAS Procedure Output," on page 187

SORT

Sorts observations in a SAS data set by one or more variables, storing the resulting sorted observations in a new SAS data set or replacing the original data set

Language element: procedure

OpenVMS specifics: available sort routines

Syntax

PROC SORT <*option(s)*> <*collating-sequence-option*>;

Note: This is a simplified version of the SORT procedure syntax. For the complete syntax and its explanation, see the SORT procedure in *SAS Procedures Guide*. △

option(s)

NODUPKEY
under OpenVMS, the observation that is returned is unpredictable; that is, the observation returned is not guaranteed to be the first observation that was encountered for that BY variable. For further explanation of the NODUPKEY option, see "NODUPKEY Option" on page 349.

SORTWKNO=*n*
specifies the number of sort work files to be used by the OpenVMS sort utility. The value for *n* can be 0 through 6. For further explanation of the SORTWKNO= option, see "SORTWKNO= Option" on page 349.

Details By default under OpenVMS, the SORT procedure uses the ASCII collating sequence. Whenever the SORT procedure uses the HOST sort utility, it uses the OpenVMS sort utility. (For information about how the sort utility is chosen, see the discussion of the SORTPGM= system option in "SORTPGM=" on page 437.) The HOST sort utility accepts all options that are available to the SAS sort utility. For a complete list of options, see the SORT procedure in the *SAS Procedures Guide*.

NODUPKEY Option The SAS sort utility and the OpenVMS sort utility differ slightly in their implementation of the NODUPKEY option. If you need to use both the NODUPKEY and EQUALS options (that is, if you need to guarantee that the first observation returned is the first observation that was input), then use the SAS sort utility.

When you use the SAS sort utility, the NODUPKEY option implies the EQUALS option by default. As a result, the observation that is returned for like BY values is the first observation that was encountered for the key BY variable. That is, the observations are returned in the order in which they were input.

By contrast, the OpenVMS sort utility does not support the EQUALS option in conjunction with the NODUPKEY option. When NODUPKEY is used with the OpenVMS sort utility, the EQUALS option is set to NOEQUALS unconditionally. As a result, when NODUPKEY is specified with the OpenVMS sort utility, the observation that is returned for observations with like BY values is not guaranteed to be the first observation that was encountered for that BY variable. The observation that the OpenVMS sort utility returns when NODUPKEY is in effect is unpredictable.

SORTWKNO= Option The SORT procedure also supports the SORTWKNO= option in the PROC SORT statement. The SORTWKNO= option specifies the number of sort work files to be used by the OpenVMS sort utility.

The OpenVMS sort utility can support up to 10 work files. If you set SORTWKNO= to 0 and define the ten sort work files, the SAS System uses the ten files. To use the sort work files, you must define a SORTWORK# logical name for each sort work area. For example:

```
$DEFINE SORTWORK0  DISK1:[TEMP]
$DEFINE SORTWORK1  DISK2:[TEMP]
$DEFINE SORTWORK2  DISK3:[TEMP]
```

The following example uses the SORTWKNO= option to specify that four work files should be used:

```
libname mylib '[mydata]';
```

```
proc sort data=mylib.june sortwkno=4;
   by revenue;
run;
```

Customizing Collating Sequences The options EBCDIC, ASCII, NATIONAL, DANISH, SWEDISH, and REVERSE specify collating sequences that are stored in the HOST catalog.

If you want to provide your own collating sequences or change a collating sequence provided for you, use the TRANTAB procedure to create or modify translation tables. For complete details on the TRANTAB procedure, see *SAS Procedures Guide*. When you create your own translation tables, they are stored in your PROFILE catalog, and they override any translation tables that have the same names in the HOST catalog.

Note: System managers can modify the HOST catalog by copying newly created tables from the PROFILE catalog to the HOST catalog. Then all users can access the new or modified translation table. △

If you are using the SAS windowing environment and want to see the names of the collating sequences that are stored in the HOST catalog, issue the following command from any window:

```
CATALOG SASHELP.HOST
```

If you are not using the SAS windowing environment, then issue the following statements to generate a list of the contents of the HOST catalog:

```
proc catalog catalog=sashelp.host;
contents;
run;
```

Entries of type TRANTAB are the collating sequences.

To see the contents of a particular translate table, use the following statements:

```
proc trantab table=table-name;
list;
run;
```

The contents of collating sequences are displayed in the SAS log.

See Also

- SORT procedure in *SAS Procedures Guide*
- TRANTAB procedure in *SAS Procedures Guide*
- System option: “SORTPGM=” on page 437
- “Working with Grouped or Sorted Observations” in *SAS Language and Procedures: Usage*

VAXTOAXP

Converts the format of a SAS data set that was created in an OpenVMS VAX environment to the format that SAS supports in the OpenVMS Alpha environment

Language element: procedure

OpenVMS Alpha specifics: All aspects are specific to the OpenVMS Alpha operating environment

Syntax

PROC VAXTOAXP

DATA = <*libref.*>*member*

OUT = <*SAS-data-set*>;

DATA=*libref.member*
specifies a libref member, a fully qualified OpenVMS pathname (in quotes), or an OpenVMS logical name. The DATA= option is required.

OUT=*SAS-data-set*
names the SAS data set that is created to hold the converted data. The OUT= option is optional. If you do not specify a value for the OUT= option, then SAS creates a temporary SAS data set called WORK.DATA*n*.

Details The only statement that is associated with the VAXTOAXP procedure is PROC VAXTOAXP.

In the OpenVMS VAX operating environment, the SAS System stores numeric variables as D-floating data types, which means that their length varies from 2 to 8 bytes. However, in the OpenVMS Alpha operating environment, numeric variables are stored as IEEE T-floating data types, which means that their length varies from 3 to 8 bytes. If you attempt to move data with 2-byte numeric variables from an OpenVMS VAX environment to an OpenVMS Alpha environment, you will get the following error message:

ERROR: IEEE numbers with a length less than 3 are not supported.
This data set contains observations with numeric variables of length 2. The data set cannot be created/translated.

If a SAS data set that was created in the OpenVMS VAX environment contains only numeric variables with lengths of 3 bytes or greater, then the SAS System in the OpenVMS Alpha environment can access the data set without the need for any conversion process. However, if your OpenVMS VAX data set does contain numeric variables with 2-byte lengths, your OpenVMS Alpha environment will be unable to access the data set until you have converted it.

The VAXTOAXP procedure increases the length of all numeric variables by 1 byte up to 8 bytes. Thus, a 2-byte numeric variable becomes 3 bytes, a 3-byte numeric variable becomes 4 bytes, and so on.

If you run the VAXTOAXP procedure on a data set that does not contain numeric variables with lengths less than 8 bytes, the conversion proceeds after the following warning message is issued:

WARNING: No numeric variables had a length less than 8, so it was unnecessary to invoke PROC VAXTOAXP.
However, the data set will still be copied as requested.

Example

Suppose you have a permanent SAS data set named CHARLIE that you created in an OpenVMS VAX environment. You know that this data set contains numeric variables with 2-byte lengths. To read that data set into the SAS System running in an OpenVMS Alpha environment, use the following statements:

```
libname vlib v6 'user$disk:[dir]';
libname alib v8 '[nwdir]';

proc vaxtoaxp data=vlib.charlie out=alib.charlie;
run;
```

You can verify that all 2-byte numeric variables have been converted to 3 bytes by running the CONTENTS procedure on the ALIB.CHARLIE data set. All numeric variables that have lengths less than 8 bytes will have their lengths increased by 1 byte.

Note: If you attempt to use the VAXTOAXP procedure while running in an OpenVMS VAX operating environment, you will receive the following error message: **ERROR: Procedure VAXTOAXP is not supported on the VAX.** △

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

SAS® Companion for the OpenVMS Environment, Version 8

1–58025–526–4

1st printing, October 1999

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

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